

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application:

2000年 1月18日

出願番号
Application Number:

特願2000-013957

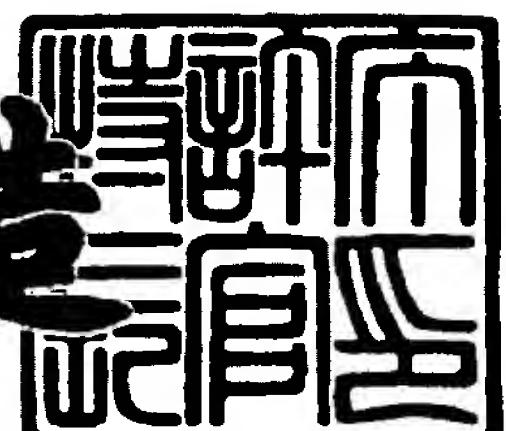
出願人
Applicant(s):

ソニー株式会社

2000年11月17日

特許庁長官
Commissioner,
Patent Office

及川耕造



出証番号 出証特2000-3095276

【書類名】 特許願
【整理番号】 9900691003
【提出日】 平成12年 1月18日
【あて先】 特許庁長官殿
【国際特許分類】 H04L 12/40
【発明者】
【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社
内
【氏名】 堀口 麻里
【特許出願人】
【識別番号】 000002185
【氏名又は名称】 ソニー株式会社
【代表者】 出井 伸之
【代理人】
【識別番号】 100067736
【弁理士】
【氏名又は名称】 小池 晃
【選任した代理人】
【識別番号】 100086335
【弁理士】
【氏名又は名称】 田村 榮一
【選任した代理人】
【識別番号】 100096677
【弁理士】
【氏名又は名称】 伊賀 誠司
【手数料の表示】
【予納台帳番号】 019530
【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1
【物件名】 図面 1
【物件名】 要約書 1
【包括委任状番号】 9707387
【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 情報処理装置及び方法、媒体

【特許請求の範囲】

【請求項1】 ネットワークを介して他の情報処理装置と接続可能であるとともに、所定の機能を実行する1以上の機能実行手段と、上記機能実行手段の使用予定に関する第1の情報を格納する格納手段とを少なくとも有する情報処理装置において、

上記機能実行手段の使用予定に関する第1の情報に含まれない第2の情報を入手する入手手段と、

上記第2の情報を所定のブロック形式とし、上記第1の情報の付加情報として上記格納手段に格納させる制御手段とを有することを特徴とする情報処理装置。

【請求項2】 上記第1の情報は、所定のデジタルインターフェイススマートにて規定された上記使用予定の情報であり、

上記第2の情報は、上記使用予定の詳細を表す表示可能なテキスト情報であることを特徴とする請求項1記載の情報処理装置。

【請求項3】 上記第2の情報は、ローテキストと当該ローテキストのコーディングを表すキャラクタコード及び言語コードを含むことを特徴とする請求項2記載の情報処理装置。

【請求項4】 上記制御手段は、上記ネットワークに接続された他の情報処理装置の上記格納手段に対して、上記第1の情報及び第2の情報の書き込みと読み出しを制御する機能を有することを特徴とする請求項1記載の情報処理装置。

【請求項5】 上記制御手段は、上記ネットワークに接続された他の情報処理装置の制御手段からの指令に基づいて、上記格納手段に対する上記第1の情報及び上記第2の情報の書き込みと読み出しを制御する機能を有することを特徴とする請求項1記載の情報処理装置。

【請求項6】 上記入手手段は、装置の使用者から入力された上記機能実行手段の使用予定に関する設定情報から上記第2の情報を生成する機能を有することを特徴とする請求項1記載の情報処理装置。

【請求項7】 上記入手手段は、上記ネットワークに接続された他の情報処理装置の格納手段に格納されている上記第2の情報を入手する機能を有することを特徴とする請求項1記載の情報処理装置。

【請求項8】 ネットワークを介して他の情報処理装置と接続可能であるとともに、所定の機能を実行する1以上の機能実行手段と、上記機能実行手段の使用予定に関する第1の情報を格納する格納手段とを少なくとも有する情報処理装置の情報処理方法において、

上記機能実行手段の使用予定に関する第1の情報に含まれない第2の情報を入手し、

上記第2の情報を所定のブロック形式とし、上記第1の情報の付加情報として上記格納手段に格納させる制御を行うことを特徴とする情報処理方法。

【請求項9】 上記第1の情報は、所定のデジタルインターフェイススマートにて規定された上記使用予定の情報であり、

上記第2の情報は、上記使用予定の詳細を表す表示可能なテキスト情報であることを特徴とする請求項8記載の情報処理方法。

【請求項10】 上記第2の情報は、ローテキストと当該ローテキストのコーディングを表すキャラクタコード及び言語コードを含むことを特徴とする請求項9記載の情報処理方法。

【請求項11】 上記ネットワークに接続された他の情報処理装置の上記格納手段について、上記第1の情報及び第2の情報の書き込みと読み出しを制御することを特徴とする請求項8記載の情報処理方法。

【請求項12】 上記ネットワークに接続された他の情報処理装置からの指令に基づいて、上記格納手段に対する上記第1の情報及び上記第2の情報の書き込みと読み出しを制御することを特徴とする請求項8記載の情報処理方法。

【請求項13】 装置の使用者から入力された上記機能実行手段の使用予定に関する設定情報から上記第2の情報を生成することを特徴とする請求項8記載の情報処理方法。

【請求項14】 上記ネットワークに接続された他の情報処理装置の格納手段に格納されている上記第2の情報を入手することを特徴とする請求項8記載の情報処理方法。

【請求項15】 1以上の所定の機能の使用予定に関する第1の情報に含まれない第2の情報を入手するステップと、

上記第2の情報を所定のブロック形式とし、上記第1の情報の付加情報として格納手段に格納させるステップとを含むことを特徴とするプログラムを情報処理装置に実行させる媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、例えばIEEE1394シリアルデータバスを介して他の情報処理装置と接続される情報処理装置において、内蔵するサブユニットを重複することなく確実に制御することができるようとする情報処理装置及び方法、媒体に関するもの。

【0002】

【従来の技術】

近年は、例えばIEEE (The Institute of Electrical and Electronics Engineers) で規格化されたIEEE1394シリアルデータバスを用いるネットワークを介して、相互に情報を伝達できるようにしたAV機器が開発されている。このネットワークシステムにおいては、所定のデジタルインターフェイスコマンド (AV/C Command Transaction Set: 以下AV/Cコマンドと略称する) を用いて、上述のネットワークに接続されているAV機器を相互に制御することが可能である。

【0003】

図34には、IEEE1394シリアルデータバスを用いたネットワークの一例を示す。この図34のネットワークシステムにおいては、IEEE1394シリアルデータバス80 (以下、単にバス80とする) を用いることにより、例えばデジタル衛星放送を受信するIRD (Integrated Reciever Decoder) 71

で受信された映像信号を、バス80介して接続されているDVCR (Digital Video Cassette Recorder) 81で録画することができる。さらにこれらのIRD71、DVCR81を用いて、いわゆる予約録画をすることも可能である。

【0004】

この装置で予約録画を行う場合には、例えばIRD71内に設けられるコントローラ72によってIRD71自身とDVCR81とが制御される。すなわち、予約録画の設定（チャンネル、開始時刻等の設定）は、IRD71に対して行われる。そして、設定された開始時刻になると、コントローラ72は、IRD71内のデジタルチューナサブユニット73に対して内部的に制御し、上記予約されている（設定されている）チャンネルを選局させ、CSアンテナ74で捉えた信号の中から当該デジタルチューナサブユニット73で受信した映像信号等をバス80を介してDVCR81に出力させる。また同時に、コントローラ72からは、バス80を介してDVCR81内に設けられるVCRサブユニット84に対して録画開始のコマンドが送信される。このときのVCRサブユニット84は、コントローラ72から送信されてきた録画開始コマンドに対応して、デジタルチューナサブユニット73から送信されてきた映像信号を図示しない磁気テープ等の記録媒体へ記録する。以上により、図34のネットワークにおける予約録画が行われる。

【0005】

また、DVCR81もコントローラ82を備えており、当該コントローラ82は、例えば内蔵のアナログチューナサブユニット83で受信された映像信号をVCRサブユニット84で記録するなどの制御を行う。

【0006】

なお、バス80に接続されているIRD71およびDVCR81のような電子機器はユニットと呼ばれており、ユニット間においては、AV/Cコマンド(AV/C Command Transaction Set)の一般的な仕様(AV/C Digital Interface Command Set General Specification、以下、AV/Cジェネラルと略称する)で規定されている記述(Descriptor、以下、適宜ディスクリプタと呼ぶ)を用いて、各ユニットに記憶されている情報を相互に読み書きすることが可能となされている

。AV/Cジェネラルの詳細については、「<http://www.1394ta.org/>」に公開されている。また、ユニットが有する機能は、サブユニットと呼ばれており、図34の例では、IRD71のディジタルチューナサブユニット73、DVCR81のVCRサブユニット84等がサブユニットとなる。

【0007】

【発明が解決しようとする課題】

ところで上述したように、DVCR81の動作を、バス80を介して接続されている他の機器（図34の例の場合、IRD71）が制御できるようなネットワークシステムの場合は、いわゆるダブルブッキングが生ずる虞がある。

【0008】

例えば、デジタル衛星放送の録画予約（録画予約Aとする）をIRD71に入力すると、その予約情報は、IRD71のコントローラ72に記憶されることになる。但し、この時点（録画予約の時間になる前の時点）では、IRD71からDVCR81に対してその録画予約に関連したAV/Cコマンドが送信されていないため、当該DVCR81は、IRD71に録画予約の設定がなされていることを知らない。その後、例えば録画予約Aの録画時刻に重複する時刻において放送される地上波アナログ放送の録画予約（録画予約Bとする）がDVCR81に設定された場合、当該DVCR81のコントローラ82は、IRD71に設定されている録画予約Aに関する情報を得ていないので、時刻が重複しているか否かを知ることができず、上記録画予約Bを受け付けて記憶してしまう。

【0009】

したがって、録画予約Aおよび録画予約Bで指定された時刻になると、DVCR81のVCRサブユニット84には、IRD71のディジタルチューナサブユニット73及びDVCR81のアナログチューナサブユニット83の両方から映像信号が供給されると共に、IRD71のコントローラ72及びDVCR81のコントローラ82の両方による制御がなされてしまうことになり、不都合が生じてしまう。

【0010】

上述した従来のネットワークシステムにおいて、このようなダブルブッキング

による不都合が発生するのは、例えば、バス80を介して接続されている各AV機器（ユニット）が他のAV機器による予約情報等の詳細入手できること、或いは、各AV機器が他のAV機器に対して自己が持つ予約情報等の詳細を知らせる手段がないこと、また、ユーザが新たな予約の入力を行う場合に既に設定されている予約情報の詳細を知り得ないこと、更には、各AV機器やユーザが予約情報の詳細を入力したくても入力できないことなどに起因している。

【0011】

そこで、本発明はこのような状況に鑑みてなされたものであり、ユーザや各機器が予約情報等の詳細を入力できること、各機器やユーザが他の機器による予約情報等の詳細入手できる（知り得る）こと、及び、各機器が他の機器やユーザに対して自己が持つ予約情報等の詳細を知らせる手段を実現することにより、ダブルブッキングの発生を抑止可能となるだけでなく、分かり易く且つ使い易いネットワークシステムを構築可能とする、情報処理装置及び方法、媒体を提供することを目的とする。

【0012】

【課題を解決するための手段】

本発明の情報処理装置は、上述した課題を解決するために、ネットワークを介して他の情報処理装置と接続可能であるとともに、所定の機能を実行する1以上の機能実行手段と、上記機能実行手段の使用予定に関する第1の情報を格納する格納手段とを少なくとも有する情報処理装置において、上記機能実行手段の使用予定に関する第1の情報に含まれない第2の情報を入手する入手手段と、上記第2の情報を所定のブロック形式とし、上記第1の情報の付加情報として上記格納手段に格納させる制御手段とを有することを特徴とする。

【0013】

本発明の情報処理方法は、上述した課題を解決するために、ネットワークを介して他の情報処理装置と接続可能であるとともに、所定の機能を実行する1以上の機能実行手段と、上記機能実行手段の使用予定に関する第1の情報を格納する格納手段とを少なくとも有する情報処理装置の情報処理方法において、上記機能実行手段の使用予定に関する第1の情報に含まれない第2の情報を入手し、上記

第2の情報を所定のブロック形式とし、上記第1の情報の付加情報として上記格納手段に格納させる制御を行うことを特徴とする。

【0014】

本発明の媒体は、上述した課題を解決するために、1以上の所定の機能の使用予定に関する第1の情報に含まれない第2の情報を入手するステップと、上記第2の情報を所定のブロック形式とし、上記第1の情報の付加情報として格納手段に格納させるステップとを含むことを特徴とするプログラムを情報処理装置に実行させることを特徴とする。

【0015】

【発明の実施の形態】

本発明の好ましい実施の形態について、図面を参照しながら説明する。なお、本明細書において、システムの用語は、複数の装置、手段などにより構成される全体的な装置を意味するものである。

【0016】

図1には、本発明実施の形態のネットワークシステムの概略構成を示す。

【0017】

この図1に示すネットワークシステムは、IEEE1394シリアルデータバス2（以下、バス2とする）を介して接続されているIRD1及びDVCR（例えばD-VHS等のレコーダ）3から構成されている。もちろん、このバス2にはIRD1やDVCR3以外に、例えば、パーソナルコンピュータ、ハードディスクドライブ、CDプレーヤ、モニタ、デジタルビデオカメラ、MD（商標）プレーヤ等のIEEE1394端子を備える電子機器を接続することが可能である。

【0018】

IRD1のコントローラ11は、ユーザからの選局操作や録画予約操作等を受け付けて、IRD1の全体を制御する。また、当該コントローラ11は、所定のデジタルインターフェイスコマンドとして、前記AV/Cコマンドを用いてDVCR3を制御する。CSアンテナ13は、図示せぬ通信衛星を介して送信されてくるデジタル衛星放送のデジタル信号を受信して、チューナサブユニット

12に出力する。チューナサブユニット12は、コントローラ11の制御に基づいて、CSアンテナ13から入力されたデジタル信号から所望のチャンネルの信号を抽出し、バス2を介してDVCR3のVCRサブユニット33に出力する。さらに、コントローラ11は、DVCR3のBBS(Bulletin Board Subunit)34に記録されている情報を検索可能となっている。

【0019】

IRD1のサブユニットの一つであるBBS14は、コントローラ11が受け付けて、確定した録画予約等の情報（詳細は図2により後述する）を格納する。

【0020】

DVCR3のコントローラ31は、ユーザからの再生指示の操作や録画予約操作等を受け付けて、DVCR3の全体を制御する。また、コントローラ31は、BBS34に記録されている情報を検索可能である。アナログチューナサブユニット32は、コントローラ31の制御に基づいて、入力されるアナログ信号から所定のチャンネルの信号を抽出し、VCRサブユニット33に出力する。

【0021】

VCRサブユニット33は、アナログチューナサブユニット32から入力された映像信号、又は、バス2を介して入力されるIRD1のチューナサブユニット12からの映像信号を図示せぬ磁気テープに記録する。

【0022】

BBS34は、DVCR3に関する録画予約等の情報（詳細は図2により後述する）を格納する。

【0023】

この図1に示したネットワークシステムにおいて、デジタル衛星放送の録画予約を行う場合、ユーザは、IRD1に対して録画予約の設定（チャンネル及び録画開始時刻等の設定）を入力する。そして、その録画予約の時刻やチャンネル等が既に予約されているものでない場合、当該入力された録画予約が認められて、その情報がIRD1のBBS14に書き込まれることになる。

【0024】

ここで、IRD1に対してユーザにより録画予約の入力がなされた場合、当該

IRD1のコントローラ11は、図2に示すように、それぞれの録画予約についての内容を表す設定情報としてスケジュールアクション（Scheduled Action 1～n）51を内部のレジスタに保持する。当該スケジュールアクション（Scheduled Action）は、スケジュールデータ（Scheduled Data）52とプログラム（Program）53からなる。また、スケジュールデータ（Scheduled Data）52は、録画の開始時間情報（Start time）、実行時間情報（Duration）、繰り返し情報（Repeat information）、使用するサブユニットの情報（Resources used）からなり、プログラム（Program）53は、サブユニットを実際に制御するためのコマンド（Command 1～n）からなる。上記スケジュールデータ（Scheduled Data）52の内容（Resource Schedule Object）は、ユーザから入力された録画予約が確定した時に、IRD1のコントローラ11からDVCR3へ送られ、当該DVCR3のBBS34に書き込まれる。一方、上記プログラム（Program）53の各コマンドは、実際に録画予約の開始時間になり且つ実行時間になった時に、IRD1のコントローラ11からDVCR3のサブユニット（録画を行う際に実際に使用される各サブユニットA, B, …, n）に送られることになる。

【0025】

DVCR3のBBS34は、図2に示すように、RSB（Resource Schedule Board）61、及びその他のボード62から構成される。RSB61は、BBS34内において録画予約の情報のように、リソースの使用予定を書き込むためのリストである。なお、当該RSB61で読み書きされるエントリは使用予定であって、予約された制御を実行するものではない。RSB61には、上記IRD1のコントローラ11より送られてきた上記スケジュールデータ（Scheduled Data）52のオブジェクト（Start time, Duration, Repeat info, Resources）や、他のユニット（DVCR3のコントローラ31も含む）より送られてきた各スケジュールデータのオブジェクトが、リソーススケジュールエントリ（Resource Schedule Entry 1～n）63として書き込まれる。

【0026】

なお、DVCR3のBBS34に書き込まれた情報は、IRD1のコントローラ11だけでなく、他のユニットのコントローラ（例えば、DVCR3のコント

ローラ31も含む)からの要求に対応しても公開される。

【0027】

図3には、BBSのRSBのフォーマット構成を示す。

【0028】

この図3において、`descriptor_length`はRSBの長さを表す。`list_type`には、RSBが読み出し専用(Read Only)であるのか又は書き込み可能(Write Enabled)であるのかが記述される。`size_of_list_specific_information`は、`list_specific_information`の長さを表し、`list_specific_information`は、`list_type`によって異なるものとなる。

【0029】

`Write Enabled list_specific_infomation`には、図4に示す情報が記述される。`Write Enabled list_specific_infomation`の`non_info_block_fields_length`は、`non info block fields`のバイト数を表す。RSBの場合、`board_type`は図5に示すように”`0116`”とされる。`object_list_maximum_size`は、`object list`の最大の大きさを表す。`object_entry_maximum_number`は、`list`における`object entries`の最大の数を表す。`object_entry_maximum_size`は、`object entry`の最大の大きさを表す。以上の`object_list_maximum_size`, `object_entries_maximum_number`, `object_entry_maximum_size`は、それぞれ制限が設けられていない場合には、”`000016`”とされる。これらの3つのフィールドは、コントローラが`object list`または、`object entry`の容量を知る上において、有意義である。`board_type_dependent_information_length`は、`board_type_dependent_information`の長さを表し、`board_type_dependent_information`は、`board type`に固有の情報を表す。

【0030】

`objec_entry`は図6に示すように記述される。`objec_entry`の`descriptor_length`は、この`descriptor`の長さを表す。`entry_type`は`Board entry Descriptor`の場合、`Board`を表す値”`8016`”とされる。`object_ID`は、`posting_device_GUID`と`record_ID`とにより構成される。`posting_device`は、BBSに対して情報を記入(post)したコントローラを意味し、従って`posting_device_GUID`は、そのGUIL

Dを表す。record_IDは、ユニット内においてイベント(Event)に対して割当てられたIDを表す。size_of_entry_specific_informationは、entry_specific_informationとしてのResource Schedule Entryの大きさを表す。

【0031】

Resource Schedule Entryには、図7に示す情報が記述される。

【0032】

この図7において、non_info_block_lengthは、repeat_informationまでのnon info block fieldsのバイト数を表す。start_timeは、図8に示すように、イベント(event、この例では予約録画)が開始する年月日時分秒を表す。年(year)は16ビットで表され、西暦を表す4つの数字がそれぞれ4ビットのBCD(Binary Coded Decimal)で表される。月(month)は8ビットで表され、1(01)月から12月までの2桁の数字を表す2つの数字がそれぞれ4ビットのBCDで表される。日(day)は8ビットで表され、1(01)日から31日までの2桁の数字(日)を表す2つの数字がそれぞれ4ビットのBCDで表される。時間(hour)は8ビットで表され、0(00時)から24時までの2桁の数字(時刻)を表す2つの数字がそれぞれ4ビットのBCDで表される。分(minute)は8ビットで表され、0(00)分から60分までの2桁の数字(分)を表す2つの数字がそれぞれ4ビットのBCDで表される。秒(second)は8ビットで表され、0(00)秒から60秒までの2桁の数字(秒)を表す2つの数字がそれぞれ4ビットのBCDで表される。このように、start_timeがBCDで表されることにより、その識別が容易となっている。またこの時刻は、ローカルタイムで表される。

【0033】

イベント(event)の長さを表すDurationは、図9に示すように、時分秒で表される。時間(hours)は3桁の時間を表す3つの数字がそれぞれ4ビットのBCDで表され、合計12ビットとなる。分(minutes)は2桁の分を表す2つの数字がそれぞれ4ビットのBCDとされ、合計8ビットとなる。秒(seconds)は2桁の秒を表す2つの数字がそれぞれ4ビットのBCDで表され、合計8ビットとされる。start_timeにDurationを加算することで、イベントの終了時刻が表

される。このように、終了時刻を直接的に表現せずに、Durationとしてイベントの長さを表すようになると、イベントのstart_timeが変更されたような場合においても、その終了時刻を変更するような操作が不要となり、変更処理が容易となる。

【0034】

`repeat_information_length`は、`repeat_information`の長さを表している。`repeat_information`は、いつどのようにしてスケジュール（schedule）が繰り返される（この場合は、予約録画が繰り返される）かを表す。Scheduled Actionが繰り返されない場合、`repeat_infomation_length`は、”00₁₆”とされる。`repeat_information`は、選択されたrepeat typeによって内容が異なる。

【0035】

`repeat_type`には、図10に示すように、`Weekly schedule`の”00₁₆”と`Interval schedule`の”10₁₆”とがある。スケジュール（schedule）が、週毎に繰り返される場合、ポスティングデバイス（Posting Device、この例ではIRD1のコントローラ11）は、その曜日及び繰り返されるべきイベントの数を図11に示すように表示する。この`repeat_type`には、図10に示す値、”00₁₆”が記述される。`number_of_events`には、イベントの数が記述される。日曜日から土曜日までの`Weekly flags`は、繰り返されるイベントが開始する週の日を表す。例えば、13時00分から3時間のイベント（この例では予約録画）が、毎週月曜日と水曜日に開始される場合、月曜日と水曜日のフラグに”1”が設定され、その他のフラグには”0”が設定される。このように、`repeat_type`には、週毎に繰り返されるイベントが記録できるので、例えば月曜日と水曜日の絶対的な日時を放送日に対応する分だけ記憶させるような場合に比べて、記憶容量は小さく済ませることが可能となる。

【0036】

ポスティングデバイス（この例ではコントローラ11）は、Scheduled Actionが所定の間隔（intervals）で繰り返される場合、図12に示すようなフォーマットでイベントを記述する。この図12の`repeat_type`には、この例の場合、図10に示す値”10₁₆”が記述される。`number_of_events`には、イベントの数

が記述される。`interval`（間隔）は、現在のイベントの`start_time`から次のイベントの`start_time`までの間隔を表す。この間隔は、時分秒で表される。時間は3桁の時間を表す3つの数字がそれぞれ4ビットのBCDで表され、合計12ビットで表現される。分と秒は、それぞれ2桁の分と秒を表す2つの数字が4ビットのBCDとされ、それぞれ合計8ビットで表現される。このように、`repeat_type`には、周期的に繰り返されるイベントが記憶できるので、それが開始される絶対時刻（日時）を別に記憶させる場合に比べて、記憶容量が少なくて済む。

【0037】

さらに、図7に示すinfo Blocksは、図13に示すようなフォーマットとされる。

【0038】

この図13において、`compound_length`は、このinfo blockのバイト長を表す。ただし`length`フィールド自身は、この長さに含まれていない。`info_block_type`は”8900₁₆”にセットされる。`primary_fields_length`は、`number_of_subunits`と`subunit_type_and_ID field`のバイト数を表す。`number_of_subunits`は、ポスティングデバイス（この例ではコントローラ11）が使用するサブユニットの数を表す。`subunit_type_and_ID`は、ポスティングデバイスが使用するサブユニットを指定する。

【0039】

また、本実施の形態の場合、図7に示すResource Schedule Entryには、図14に示す`character_code_information_block`と、図15に示す`language_code_information_block`と、図16に示す`raw_text_information_block`とが記述される。

【0040】

図14に示す`character_code_information_block`には、図16の`raw_text_information_block`の`raw_text_data`が何れのキャラクタコード（character code）でコーディングされているのかを示す情報が記述されている。この図14において、`compound_length`は、この`character_code_information_block`のバイト長を表す。ただし、`length`フィールド自身は、この長さに含まれていない。`info_blo`

`ck_type`は`character_code_information_block`であることを示す” $0\ 0\ 0\ 8_{16}$ ”にセットされる。`primary_fields_length`は、`character_code_type`と`character_code_type_specific field`のバイト数を表す。`character_code_type`はキャラクタコード (`character code`) の種類を表し、`character_code_type_specific`はキャラクタコードの仕様を表す。

【0041】

図15に示す`language_code_information_block`には、図16の`raw_text_info`
`mation_block`の`raw_text_data`が何れの言語コード (`language code`) でコーディングされているのかを示す情報が記述されている。この図15において、`compound_length`は、この`language_code_information_block`のバイト長を表す。ただし、`length`フィールド自身は、この長さに含まれていない。`info_block_type`は”`language_code_information_block`であることを示す” $0\ 0\ 0\ 9_{16}$ ”にセットされる。`primary_fields_length`は、`language_code_type`と`language_code_type_specific field`のバイト数を表す。`language_code_type`は言語コード (`language cod`e) の種類を表し、`language_code_type_specific`は言語コードの仕様を表す。

【0042】

図16に示す`raw_text_information_block`において、`compound_length`は、この`raw_text_information_block`のバイト長を表す。ただし、`length`フィールド自身は、この長さに含まれていない。`info_block_type`は`raw_text_information_block`であることを示す” $0\ 0\ 0\ A_{16}$ ”にセットされる。`primary_fields_length`は、`raw_text_data`のバイト数を表す。`raw_text_data`には、図17に示すように、録画予約に関連する情報として、例えば、チャンネル、プログラム名（番組名）、制御情報（再生、録画、停止等を表すテキスト）、備考（例えばペイパービューの番組等）、プロバイダ名、予約が仮予約である旨の情報などのテキストデータが記述される。

【0043】

図1に示した本発明実施の形態のネットワークシステムにおいては、各ユニットのRSBのResource Schedule Entryにテキスト情報を書き込むためのinformation blockとして、`character_code_information_block`と、`language_code_info`

`rmation_block`と、`raw_text_information_block`とを定義しており、これらを用いて、例えば図17に示したような録画予約に関する情報など、機器の使用予定に関する様々な付加情報をテキストデータとして書き込み可能とし、また、このテキストデータはネットワークシステムを構成する各ユニットのコントローラにおいて読み取り可能となっている。

【0044】

したがって、本実施の形態のネットワークシステムにおいては、ユーザや各機器が上記録画予約に関する情報を入力できること、各機器やユーザが他の機器による上記録画予約に関する情報を入手できる（知り得る）こと、及び、各機器が他の機器やユーザに対して自己が持つ上記録画予約に関する情報を知らせることなどが実現可能となっている。

【0045】

図18～図20には、本実施の形態のネットワークシステムにおいて上記録画予約に関する情報の入力、入手、録画予約に関する情報の報知を行う際の手順を示す。なお、これら図18～図20の処理は、録画予約に関するリソース、すなわちサブユニットが複数存在する場合、それら各サブユニット毎に順番に行われる処理である。

【0046】

図18に示すフローチャートのステップS10からステップS17までは、使用したい機器のRSBが書き込み可能かどうか調べる手順である。図1のネットワークシステムの例の場合、録画予約に使用するサブユニットは、IRD1のチューナサブユニット12とDVCR3のVCRサブユニット33であるため、IRD1のコントローラ11は、先ず、IRD1のBBS14内のRSBが書き込み可能であるかどうか調べる。

【0047】

この図18において、ユーザから例えば録画日時とチャンネルの録画予約の入力がなされると、IRD1のコントローラ11は、ステップS10の処理として、その予約の情報を例えば内部レジスタに一旦保存する。

【0048】

コントローラ11は、上記予約設定入力がなされると、次のステップS11の処理として、上記予約設定の内容に応じて使用するサブユニットが存在する機器内のRSBをライトオープン(WRITE OPEN)させる処理(すなわち書き込み可能な状態にする処理)を実行する。図1の例の場合、予約録画に使用するサブユニットは、IRD1のチューナサブユニット12とDVCR3のVCRサブユニット33であり、ここでは先ず、例えばチューナサブユニット12を有する機器(ユニット)であるIRD1のBBS14内のRSBをライトオープン(書き込み可能な状態)にする。なお、VCRサブユニット33を有する機器であるDVCR3のBBS34のRSB61をライトオープンする処理は、上記IRD1のRSBに対して図18～図20の処理が行われた後になる。

【0049】

ここで図21には、コントローラ11がRSBをライトオープンする場合に出力するライトオープンコマンド(WRITE OPENコマンド)のフォーマットを示す。但し、IRD1のBBS14のRSBをライトオープンする場合、BBS14とコントローラ11とは、同じIRD1内に設けられているものであり、その間にバス2は存在していないので、この場合のコントローラ11は、BBS14に対して、バス2を介してライトオープンコマンドが供給されてきた場合と同様の状態となるようにBBS14を制御する。なお、上記IRD1のRSBに対して図18～図20の処理が行われた後、DVCR3のBBS34のRSB61をライトオープンする際には、コントローラ11から図21に示すフォーマットのライトオープンコマンド(WRITE OPENコマンド)が出力され、バス2を介してDVCR3のBBS34(実際にはBBS34を制御するコントローラ31)へ送られることになる。

【0050】

この図21に示すライトオープンコマンドは、ターゲットの所定のアドレス空間にアクセス(この例ではBBSのRSBにアクセス)するために使用される、オープンディスクリプタコマンド(OPEN DESCRIPTORコマンド)の一種である。この図21に示すライトオープンコマンドにおいて、opcodeにはオープンディス

クリプタであることを表す値” 08_{16} ”が記述され、`operand0`にはライトオープンするディスクリプタの種類を表す`descriptor_type`として、リストIDにより規定されるObject List Descriptorであることを表す値” 10_{16} ”が記述される。`operand1`と`operand2`には、アクセス先の（ライトオープンする）RSBのリストID（この例においては、“00”と“01”）が記述される。さらに`operand3`には、サブファンクション（subfunction）として、ディスクリプタを読み出したり書き込みアクセスのためにオープンするライトオープンであることを表す値” 03_{16} ”が記述される。`operand4`は、リザーブのための値”00”とされている。

【0051】

次に、コントローラ11の処理は、ステップS12に進み、IRD1のBBS14におけるRSB内の`descriptor_length`と`list_specific_information_field`の情報を読み出す。なお、DVCR3のBBS34のRSB61の読み出し処理は、上記IRD1のRSBに対して図18～図20の処理が行われた後になる。

【0052】

図22には、コントローラ11がRSBをリード(READ)する場合に出力するリードコマンド(READコマンド)のフォーマットを示す。但し、IRD1のBBS14のRSBをリードする場合、BBS14とコントローラ11とは、同じIRD1内に設けられているものであり、その間にはバス2は存在していないので、この場合のコントローラ11は、BBS14に対して、バス2を介してリードコマンドが供給されてきた場合と同様の状態となるようにBBS14を制御する。なお、上記IRD1のRSBに対して図18～図20の処理が行われた後、DVCR3のBBS34のRSB61をリードオープンする際には、コントローラ11から図22に示すフォーマットのリードコマンド(READコマンド)が出力され、バス2を介してDVCR3のBBS34(実際にはBBS34を制御するコントローラ31)へ送られることになる。

【0053】

図22に示すリードコマンドのフォーマットにおいて、先頭の`opcode`には、リードディスクリプタ(read descriptor)であることを表す値の” 09_{16} ”が記

述されている。続くoperand0には読み出すディスクリプタ(descriptor)を識別するためのdescriptor identifierが記述される。上記ステップS12における読み出しの処理の場合は、リストIDでdescriptor identifierが記述される。具体的には前記図4に示したR S Bのwrite enabled list_specific_information fieldのAddress_offsetの” 00_{16} ”乃至” $0D_{16}$ ”が記述される。read_result_statusには、ポスティングデバイスがリードコマンドを送出するときは” FF_{16} ”が記述され、ターゲット(この例ではBBSのR S B)からレスポンスとして返される場合には、読み取り結果が記述される。data_lengthには、ターゲットから読み出されるべきデータのバイト数が記述される。この値が” 00_{16} ”に設定された時、全てのリストが読み出される。addressには、読み出しを開始すべきアドレスが記述される。その値が” 00_{16} ”とされた場合、先頭から読み出しが開始される。

【0054】

次に、コントローラ11は、ステップS13の処理として、この時点でのターゲットであるIRD1のBBS14内のR S Bに対するリストの最大長の制限(図4におけるobject_list_maximum_size)、リストのエントリ数の制限(図4におけるobject_entries_maximum_number)、および各エントリの最大バイト長の制限(図4におけるobject_entry_maximum_size)を抽出する。

【0055】

次に、コントローラ11は、ステップS14の処理として、これからIRD1のBBS14内のR S Bにデータ(この場合予約情報)を記録しても、ステップS13で抽出されたリストの最大長(object_list_maximum_size)を超えないか否かを判定する。ステップS14において上記リストの最大長(object_list_maximum_size)を越えないと判定した場合、コントローラ11の処理はステップS15に進み、一方、越えていると判定した場合、コントローラ11の処理はステップS17の処理に進む。

【0056】

ステップS15の処理に進むと、コントローラ11は、ステップS13で抽出されたリストのエントリ数の制限(最大エントリ数:object_entries_maximum_n

umber) から、現在のエントリ数を引いた値が”0”より大きいか否か、すなわち、まだ記録可能なエントリが残っているか否かを判定する。当該ステップS15において大きい（記録可能なエントリが残っている）と判定された場合、コントローラ11の処理はステップS16に進み、一方、大きくない（記録可能なエントリが残っていない）と判定された場合、コントローラ11の処理はステップS17の処理に進む。

【0057】

ステップS16の処理に進むと、コントローラ11は、ステップS13で抽出された最大エントリ長（object_entry_maximum_size）から、これから書き込もうとしている予約情報のエントリ長を引いた値が”0”より大きいか否か、すなわち、書き込むべきエントリ長に、まだ余裕があるか否かを判定する。当該ステップS16において大きい（エントリ長に余裕がある）と判定された場合、コントローラ11の処理は図19のステップS18に進み、大きくない（エントリ長に余裕がない）と判定された場合、コントローラ11の処理はステップS17に進む。

【0058】

ステップS14乃至ステップS16における条件の何れか1つが満足されずにステップS17の処理に進むと、コントローラ11は、例えば「予約が一杯です」のような警告表示を図示しない表示手段上に行う。すなわち、この場合の警告表示は、IRD1のBBS14内のRSBに予約情報を書き込むだけの余裕がなくなっていることを表している。これにより、ユーザは、予約が一杯でそれ以上予約をすることができないことを知ることができる。また、IRD1のBBS14内のRSBの図7に示したResource Schedule Entryに、前述の図17に示した予約の詳細な内容を表すテキスト情報が既に記述されている場合には、そのテキスト情報の表示を行う。これにより、ユーザは、その予約の詳細な内容を知ることができる。本実施の形態において上述のように予約の詳細な内容を例えばテキスト表示することにより、ユーザは、例えばそれら予約が必要な予約かそうでないか等の判断、すなわち必ず録画したい番組であるか或いは必ずしも録画しなくてもよい番組であるか等の判断ができ、例えば必ず録画したい番組についての

予約はそのまま維持させ、一方、必ずしも録画しなくてもよい番組についてはその予約を取り消し、別の番組の予約を新たに行うなどの、適切な行動を取ることができるようになる。また例えば、予約が仮予約なので上書きしても良いという情報をテキスト情報としてエントリ時に書き込むようにしておけば、後から予約するユーザがより適切な行動、すなわち仮予約されていたものを消して新たな予約を行ったりする等の行動を取ることができるようになる。

【0059】

一方、ステップS14乃至ステップS16の条件の何れもが満足されている場合、IRD1のBBS14内のRSBに予約情報を書き込む余裕があるので、コントローラ11は、図19のステップS18以降の処理に進み、重複する時刻の予約が既になされているか否かの判定を行う。

【0060】

すなわち、コントローラ11は、先ず図19のステップS18の処理として、変数*i*を”0”に初期設定し、次にステップS19の処理として、IRD1のBBS14のRSBに記録されているエントリの数(*number_of_entries*)から変数*i*を減算した値が”0”より大きいか否か、すなわちBBS14のRSBに記録されている全てのエントリについて検索が行われたか否かを判定する。このステップS19の処理において、*number_of_entries*から変数*i*を減算した値が”0”より大きいと判定した場合、まだ検索していないエントリが存在するので、コントローラ11の処理はステップS20に進み、BBS14のRSBに掲示されている前記図6の*object_entry[i]*を読み出す（この場合は*object_entry[0]*を読み出す）。なお、当該ステップS20における読み出しも、前記図22に示したリードコマンドで行われるが、この場合における*descriptor identifier*の記述は*object position*で行われる。この*object entry[i]*には、既に登録されている予約の時刻情報（図7における*start_time, Duration*）や、その予約において、使用するサブユニットの識別情報（図13における*subunit_type_and_ID[0]*）、前記図14～図17に示したテキスト情報などが記憶されている。

【0061】

次に、コントローラ11は、ステップS21の処理として、ステップS10でユーザより入力された時刻情報 (start_time, Duration) が、ステップS20で読み出された時刻情報 (start_time, Duration) と重複しているか否かを判定する。このステップS21において、時刻が重複していると判定された場合、コントローラ11はステップS22の処理に進み、一方、重複していないと判定した場合、コントローラ11はステップS23の処理に進む。

【0062】

ステップS21にて時刻が重複していると判定してステップS22の処理に進むと、コントローラ11は、ステップS10で予約設定されたサブユニット（この場合、チューナサブユニット12）が、ステップS20で読み出したサブユニット (subunit_type_and_ID) と一致しているか否かを判定する。当該ステップS22にてサブユニットが一致していると判定した場合、コントローラ11の処理はステップS24に進み、一方、サブユニットが一致していないと判定した場合、コントローラ11の処理はステップS23に進む。

【0063】

ステップS22にてサブユニットが一致していると判定された場合は、結局、時刻とサブユニットの両方が一致していることになるので、コントローラ11はステップS24の処理として、例えば「予約が重なっています」のような警告表示を行う。これにより、ユーザは、予約が重なることを知ることができて予約のダブルブッキングの発生を防止できることになる。また、IRD1のBBS14のRSBの図7に示したResource Schedule Entryに、テキスト情報として予約の設定内容が記述されている場合は、そのテキスト情報の表示を行う。これにより、ユーザは、その重なる予約の詳細な内容を知ることができる。本実施の形態において予約の詳細な内容を例えばテキスト表示することにより、ユーザは、例えばそれら予約が必要な予約かそうでないか等の判断ができ、例えば必要な予約についてはそのまま維持させ、また必ずしも必要でない予約を取り消し、別の新たな予約を行うなどの適切な行動を取ることができるようにになる。また例えば、予約が仮予約なので上書きしても良いという情報をテキスト情報としてエントリ

時に書き込むようにしておけば、後から予約するユーザがより適切な行動、すなわち仮予約されていたものを消して新たな予約を行ったりする等の行動を取ることができるようになる。

【0064】

一方で、ステップS21において、時刻が重複していないと判定された場合、又は、時刻が一致していたとしてもステップS22においてサブユニットが一致していないと判定された場合、予約のダブルブッキングが発生する恐れはないので、コントローラ11は、ステップS23の処理として、変数*i*を1だけインクリメントした後、ステップS19に戻り、*number_of_entries*から変数*i*を減算した値が0より大きくないと判定されるまで、同様の処理を繰り返し実行する。すなわち、コントローラ11では、IRD1のBBS14のRSBに記憶されている全ての*object entry [i]*について、重複する時刻の予約がなされているか否かの検索を行う。

【0065】

また、ステップS19において、*number_of_entries*から変数*i*を減算した値が”0”より大きくないと判定された場合（全ての*object entry [i]*の検索が終了した場合）、コントローラ11の処理は、図20のステップS25に進む。なお、前述したステップS14乃至ステップS17の処理は、ステップS19の処理において、NOと判定された場合に実行するようにすることも可能である。

【0066】

図20のステップS25の処理に進むと、コントローラ11は、IRD1のBBS14のRSBの*object entry*をクリエイト（create）する。なお、DVCR3のBBS34のRSBのクリエイト処理は、上記IRD1のRSBに対して図18～図20の処理が行われた後になる。

【0067】

図23には、コントローラ11がRSBの*object entry*をクリエイト（create）する場合に出力するクリエイトコマンド（CREATEコマンド）のフォーマットを示す。但し、IRD1のBBS14のRSBをクリエイトする場合、BBS14とコントローラ11とは、同じIRD1内に設けられているものであり、その間

にはバス2は存在していないので、この場合のコントローラ11は、BBS14に対して、バス2を介してクリエイトコマンドが供給されてきた場合と同様の状態となるようにBBS14を制御する。なお、上記IRD1のRSBに対して図18～図20の処理が行われた後、DVCR3のBBS34のRSB61のobject entryをクリエイトする際には、コントローラ11から図23に示すフォーマットのクリエイトコマンド(CREATEコマンド)が出力され、バス2を介してDVCR3のBBS34(実際にはBBS34を制御するコントローラ31)へ送られることになる。

【0068】

また、図24には、図23内のsubfunction_1で指定できる値を示し、本実施の形態では”01”(create a new object and its child list)が使用される。また、図25は、図23内のsubfunction_1_specification for subfunction_1=01のフォーマットを示す。さらに図26は、図25内の各フィールド値を示した図である。図25のdescriptor_identifier_where, descriptor_identifier_what_1,_2の各フィールドに、図26に示すように、”20₁₆”, ”22₁₆”, ”11₁₆”をそれぞれ設定すると、”create a newobject and its child list”的意味となる。

【0069】

これらAV/Cコマンドにおけるクリエイトコマンドについての詳細は、IEE1394(インターネットホームページ<http://www.1394TA.org>参照)に記述されているものであり、本実施の形態中の各図は、その文献(Enhancement to the AV/C General Specification 3.0 Version 1.0 FC2や、TA Document 1999005 AV/C Bulletion Board Subunit General Specification 1.0 Draft 0.99:149)中のものを記載してある。また、ボードを構成するインフォメーションディスクリピタ(Information List Descriptor)にも書き込み可能なものと読み出し可能なものがあり、これらの区別には、リストタイプが使用される。

【0070】

なお、外部からAV/Cディスクリピタ(AV/C Descriptor)に新規に情報を書き込む方法の1つとして、例えばコントローラ(ポスティングデバイス)がタ

ターゲットに対して前述したクリエイトコマンドを発行し、当該ターゲットが情報を書き込む雛形を作った後、再度、コントローラが具体的な内容を書き込む制御を行うような方法が一般的な方法として考えられる。例えば、初めて情報を書き込む場合、コントローラは所望のリストを指定して、AV/Cディスクリプタクリエイトコマンド (AV/C Descriptor CREATE command) を発行する。このコマンドを受けたターゲットでは、AV/Cジェネラルで指定されたデータ構造の雛形に基づいたオブジェクトをターゲットの内部に作ることになる。また、AV/Cジェネラルで決められたデータ構造の雛形には、オブジェクトIDを示すフィールドがある。AV/Cディスクリプタ (AV/C Descriptor) を用いたリストでは、オブジェクトIDはターゲットが管理することになる。つまり、オブジェクトをクリエイト (CREATE) した段階で、ターゲットがそのオブジェクトを一意に指定できるIDを付け、そのIDを管理する機能をターゲットが所有することになる。

【0071】

オブジェクトIDとは、リスト内でそのオブジェクトを一意に指定するための識別番号であり、このため当該オブジェクトIDを重複しないようにする機能が管理する側に必要になる。BBS自体は情報を提供する場所であり、オブジェクトIDの管理はコントローラが持つことになる。

【0072】

ところが、サブユニットに対してクリエイトコマンドが発行された時、矛盾が生ずる虞がある。すなわち、オブジェクトをクリエイト (CREATE) した際には、コントローラが管理すべきオブジェクトIDを、ターゲットが割り振ることになるからである。また、クリエイトコマンドの発行後は、ライト制御を続けて行う必要がある。このように、処理が複数ステップにわかれていることにより、コントローラが書き込み途中に例えばバスから外されたような場合には、不完全なオブジェクトが作成されてしまう可能性がある。

【0073】

したがって、上記のような状況においては、その不完全なオブジェクトを特定し、そのようなオブジェクトができたときに、当該オブジェクトを良好に削除で

きるシステムが必要になる。

【0074】

そこで、本発明の実施の形態では、BBSへの書き込み手段を規格で規定し、不完全なオブジェクトを特定できる仕組みを用意している。

【0075】

すなわち、先ず、ターゲットは、オブジェクトID（グローバルユニークID（Global Unique ID：GUID）とrecord IDとで構成される）のうちのGUID部分に、一時的に管理する番号（例えば、全て”0”）を割り振るようにする。コントローラは、先にオブジェクト内部に情報を書き込み、正常に書き込みが終了したならば、最後にGUIDを書き換える。

【0076】

上記の手順を決めることにより、正常に書き込み作業が終了したときには、GUIDが全て”0”となっているオブジェクトはできないことになり、したがって、GUIDが全て”0”的オブジェクトは、書き込み途中で不完全となったオブジェクトと特定できることになる。

【0077】

これにより、書き込み途中のオブジェクトを一意に特定することができ、また、正常に書き込まれたオブジェクトと不完全なオブジェクトとを区別でき、さらに不完全なオブジェクト（無効なオブジェクト）を簡単に削除することが可能となる。このことにより、電子機器に設けられている有限なメモリを有効活用できるようになる。また、書き込み途中のオブジェクトの特定方法は、オブジェクトIDのGUID部分を全て”0”にするような簡単な方法なので、不完全なオブジェクトを削除するためのソフトウェアの作成も容易となる。

【0078】

なお、図20のステップS25において、上記クリエイトコマンドを利用する代わりに、インサートコマンド（INSERTコマンド）を使用することも可能である。

【0079】

次に、ステップS26の処理に進むと、コントローラ11は、IRD1のBBS

S14のRSBのentry_specific_information fieldsの部分（図2、図7）に予約内容を書き込む。すなわち、これにより、start_time, Duration, repeat_information, 使用するサブユニット（subunit_type_and_ID）などが書き込まれる。

【0080】

図27は、このような場合にコントローラ11が出力するライトディスクリプタコマンド(WRITE DESCRIPTORコマンド)のフォーマットを表している。但し、IRD1のBBS14のRSBに書き込みを行う場合、BBS14とコントローラ11とは同じIRD1内に設けられているため、この場合のコントローラ11は、BBS14に対して、バス2を介してライトディスクリプタコマンドが供給されてきた場合と同様の状態となるようにBBS14を制御する。なお、上記IRD1のRSBに対して図18～図20の処理が行われた後、DVCR3のBBS34のRSB61に書き込みを行う際には、コントローラ11から図27に示すフォーマットのライトディスクリプタコマンドが出力され、バス2を介してDVCR3のBBS34（実際にはBBS34を制御するコントローラ31）へ送られることになる。

【0081】

図27において、先頭のopcodeには、WRITE DESCRIPTORであることを表す値” $0A_{16}$ ”が記述される。operand0には、書き込み対象となるディスクリプタを識別させるためのdescriptor identifierが記述される。この記述は、object positionで行われる。以下、subfunctionとして、partial_replaceであることを表す値” 50_{16} ”が記述される。これによりパーシャルインサート(partial insert)またはパーシャルディレート(partial delete)が実行される。インサート(insert)では、descriptor_identifierで指定されるoperandにより規定される1つ前に新しいディスクリプタ(descriptor)が挿入される。デリート(delete)ではdescriptor_identifierにより規定されるディスクリプタが削除される。group_tagは、ライトディスクリプタコマンドが発行される必要があるディスクリプタ上において、分割できない更新の処理を行うために利用される。この例においては、データをディスクリプタに迅速に書き込むことを表す値” 00_{16}

” (immediate) が記述されている。replacement_data_lengthは、replacement_dataのoperandにおけるバイト数、すなわち書き込みたいデータの長さを表している。addressは、処理が行われるべき位置を表している。replacement_data_lengthの” 0 ” はパーシャルデリート (partial delete) を意味し、その場合、replacement_dataのoperandは存在しない。この場合、originalにdata_lengthは” 0 ” より大きい値となり、それが削除されるべきバイトの数を表す。original_data_lengthが” 0 ” である場合、パーシャルインサート (partial insert) 処理が行われる。この場合、replacement_data_lengthは、” 0 ” より大きい値とされ、挿入されるべきバイトの数を表している。

【0082】

次に、コントローラ11は、ステップS27の処理として、IRD1のBBS14内のRSBの図7に示したResource Schedule Entryに、テキスト情報として予約の設定内容を追加（記述）するか否か判定し、追加すると判定した場合はステップS28の処理に進み、追加しないと判定した場合はステップS31の処理に進む。

【0083】

ステップS28に進むと、コントローラ11は、先ず図28に示すSubunit Identifier Descriptor内のRSBボードタイプを読み出す。

【0084】

図28において、descriptor_Lengthは、このディスクリプタ (Descriptor) の長さを表す。generation_IDは、このBBSにおいて、どのAV/Cディスクリプタ (AV/C descriptor) フォーマットが使用されるかを表し、通常” 00₁₆ ” とされる。size_of_list_IDは、list IDのバイト数を表す。size_of_object_IDは、このobject IDのバイト数を表す。size_of_object_posionは、list内のオブジェクト (object) の位置が参照されるときに使用されるバイト数を表す。number_of_root_object_listsは、このBBSが直接関連するroot object listsの数を表す。root_object_list_id_x (x=0, 1, 2, ..., n-1) は、このBBSが関連するroot object listsのそれぞれのIDを表す。subunit_dependent_information_lengthはsubunit_dependent_informationの長さを表し、subuni

`t_dependent_information`には、このBBSが依存するフォーマット及びコンテンツに関する情報が記述される。`subunit_dependent_information`には、`non_info_bloc`に`field_ids_length`, `bulletin_board_subunit_version`, `number_of_supported_board_type (n)`, `supported_board_type_specific_of_length [0]`の他、`supported_board_type_specific_info [0]`乃至`supported_board_type_specific_info [n-1]`と、それらの長さを表す`supported_board_type_specific_of_length [0]`乃至`supported_board_type_specific_of_length [n-1]`が含まれている。さらに、BBSには、`manufacturer_dependent_information`の長さを表す、`manufacturer_dependent_length`と製造者に依存する情報を含む`manufacturer_dependent_information`が記述される。`root_object_list_id`の値は、それがRSBを表すものである場合、図29に示すように、所定の値” 1001_{16} ”とされる。このように、RSB（図29ではResource Schedule Listと記述されている）を表すIDを、所定の値に固定しておくことにより、RSBを読み出す処理が容易となる。

【0085】

また、図28の`supported_board_type_specific_information`フィールドは、図30に示すようなフォーマットとされる。この図30において、`supported_board_type`には、図5に示したRSBであることを表す値” 01_{16} ”が記述される。`supported_board_type_version`は、bulletin Board Type Specificationのバージョンの番号を表す。`implementation_profile_ID`はこのboard typeのためのprofile IDバージョンを表す。`supported_board_type_dependent_information_length`は、`supported_type_dependent_information`のバイト数を表す。`supported_board_type_dependent_information`には、各board type specificationに固有の情報が記述される。

【0086】

次に、ステップS29の処理に進み、コントローラ11は、BBS14のRSBのバージョンが1.0、すなわちRSBのResource Schedule Entryにテキスト情報を書き込むための`information block`として、前記図7及び図14～図17に示した`character_code_information_block`と、`language_code_information_`

`block`と、`raw_text_information_block`とが定義されているバージョン1.0よりも、RSBのバーションが大きいか否か判定し、大きいと判定したときはステップS30の処理に進み、大きくないと判定したときはステップS31の処理に進む。

【0087】

ステップS30の処理に進むと、コントローラ11は、ユーザからの予約設定情報の入力に応じたテキスト情報を、上記BBS14のRSBのResource Schedule Entry内のテキスト情報を書き込むためのinformation blockに追加する。

【0088】

次に、ステップS31の処理に進むと、コントローラ11は、リスト、すなわちBBS14のRSBをクローズする。図31には、RSBをクローズする場合にコントローラ11が出力するクローズコマンド(CLOSEコマンド)のフォーマットを表している。但し、IRD1のBBS14のRSBをクローズする場合、BBS14とコントローラ11とは同じIRD1内に設けられているため、この場合のコントローラ11は、BBS14に対して、バス2を介してクローズコマンドが供給してきた場合と同様の状態となるようにBBS14を制御する。なお、上記IRD1のRSBに対して図18～図20の処理が行われた後、DVCR3のBBS34のRSBのクローズを行う際には、コントローラ11から図31に示すフォーマットのクローズコマンドが出力され、バス2を介してDVCR3のBBS34(実際にはBBS34を制御するコントローラ31)へ送られることになる。

【0089】

この図31に示すクローズコマンドのフォーマットは、基本的に図21に示したライトオープンコマンドと同様のフォーマットであり、`subfunction`が、図21においては、ライトオープン(WRITE OPEN)を表す”03₁₆”とされているのに対して、図31のクローズコマンドにおいては、クローズ(CLOSE)であることを表す値”00₁₆”とされている点が異なっている。その他の構成は図21における場合と同様である。

【0090】

次にステップS32に進み、コントローラ11は、予約に関連する他のリソースが存在するか否かを判定する。この場合、これまでの処理によりIRD1のチューナサブユニット12についての予約の処理は終了しているが、DVCR3のVCRサブユニット33についての予約の処理は行われていないため、当該ステップS32において他のリソースがあると判定され、図18のステップS11の処理に戻る。

【0091】

これ以降、コントローラ11は、当該他のリソースとしてDVCR3のVCRサブユニット33についての予約の処理を実行する。すなわち、コントローラ11は、当該DVCR3のSSB34のRSB61に対して、前述同様に、ステップS11以降の処理を行うことになる。

【0092】

当該DVCR3のVCRサブユニット33についての予約処理を実行する場合のコントローラ11は、ステップS11の処理として、VCRサブユニット33を有する機器であるDVCR3のBBS34内のRSB61を、ライトオープン（書き込み可能な状態）するために、前記図21に示したライトオープンコマンドを、バス2を介してDVCR3のBBS34（実際にはBBS34を制御するコントローラ31）へ送る。

【0093】

次に、コントローラ11は、ステップS12の処理として、前記図22に示したリードコマンドをバス2を介してDVCR3のBBS34（実際にはBBS34を制御するコントローラ31）へ送り、RSB61のdescriptor_lengthとlist_specific_information_fieldの情報を読み出す。

【0094】

次に、コントローラ11は、ステップS13の処理として、DVCR3のBBS34内のRSB61に対するリストの最大長の制限（図4におけるobject_list_maximum_size）、リストのエントリ数の制限（図4におけるobject_entries_maximum_number）、および各エントリの最大バイト長の制限（図4におけるobjec

`t_entry_maximum_size)` を抽出する。

【0095】

次に、コントローラ11は、前記ステップS14～ステップS16の処理を行う。ステップS14乃至ステップS16における条件の何れか1つが満足されずにステップS17の処理に進むと、コントローラ11は、例えば「予約が一杯です」のような警告表示を図示しない表示手段上に行う。すなわち、この場合の警告表示は、DVCR3のBBS34内のRSB61に予約情報を書き込むだけの余裕がなくなっていることを表している。これにより、ユーザは、予約が一杯でそれ以上予約をすることができないことを知ることができる。また、DVCR3のBBS34のRSB61の図7に示したResource Schedule Entryに、前述の図17に示した予約の詳細な内容を表すテキスト情報が既に記述されている場合には、そのテキスト情報の表示を行う。これにより、ユーザは、その予約の詳細な内容を知ることができる。本実施の形態において上述のように予約の詳細な内容を例えばテキスト表示することにより、ユーザは、例えばそれら予約が必要な予約かそうでないか等の判断、すなわち必ず録画したい番組であるか或いは必ずしも録画しなくてもよい番組であるか等の判断ができ、例えば必ず録画したい番組についての予約はそのまま維持させ、一方、必ずしも録画しなくてもよい番組についてはその予約を取り消し、別の番組の予約を新たに行うなどの、適切な行動を取ることができるようになる。また例えば、予約が仮予約なので上書きしても良いという情報をテキスト情報としてエントリ時に書き込むようにしておけば、後から予約するユーザがより適切な行動、すなわち仮予約されていたものを消して新たな予約を行ったりする等の行動を取ることができようになる。

【0096】

一方、ステップS14乃至ステップS16の条件の何れもが満足されている場合、DVCR3のBBS34内のRSB61に予約情報を書き込む余裕があるので、コントローラ11は、図19のステップS18以降の処理に進み、重複する時刻の予約が既になされているか否かの判定を行う。すなわち、コントローラ11は、図19のステップS18の処理として変数iを初期設定し、次にステップS19の処理として、BBS34のRSB61に記録されているエントリの数（

`number_of_entries`) から変数 `i` を減算した値が”0”より大きいか否かを判定する。このステップ S 19 の処理において、`number_of_entries` から変数 `i` を減算した値が”0”より大きいと判定した場合、まだ検索していないエントリが存在するので、コントローラ 11 はステップ S 20 の処理として、DVCR3 の BBS34 の RSB61 に掲示されている前記図 6 の `object_entry[i]` を読み出す。

【0097】

次に、コントローラ 11 は、ステップ S 21 の処理として、ステップ S 10 でユーザより入力された時刻情報 (`start_time, Duration`) が、ステップ S 20 で読み出された時刻情報 (`start_time, Duration`) と重複しているか否かを判定する。

【0098】

ステップ S 21 にて時刻が重複していると判定してステップ S 22 の処理に進むと、コントローラ 11 は、ステップ S 10 で予約設定されたサブユニット (この場合、VCR サブユニット 33) が、ステップ S 20 で読み出したサブユニット (`subunit_type_and_ID`) と一致しているか否かを判定する。

【0099】

ステップ S 22 にてサブユニットが一致していると判定された場合は、結局、時刻とサブユニットの両方が一致していることになるので、コントローラ 11 はステップ S 24 の処理として、例えば「予約が重なっています」のような警告表示を行う。これにより、ユーザは、予約が重なることを知ることができて予約のダブルブッキングの発生を防止できることになる。また、DVCR3 の BBS34 の RSB61 の図 7 に示した Resource Schedule Entry に、テキスト情報として予約の設定内容が記述されている場合は、そのテキスト情報の表示を行う。これにより、ユーザは、その重なる予約の詳細な内容を知ることができる。本実施の形態において予約の詳細な内容を例えばテキスト表示することにより、ユーザは、例えばそれら予約が必要な予約かそうでないか等の判断ができる、例えば必要な予約についてはそのまま維持させ、また必ずしも必要でない予約を取り消し、別の新たな予約を行うなどの適切な行動を取ることができるようになる。また例

えば、予約が仮予約なので上書きしても良いという情報をテキスト情報としてエントリ時に書き込むようにしておけば、後から予約するユーザがより適切な行動、すなわち仮予約されていたものを消して新たな予約を行ったりする等の行動を取ることができるようになる。

【0100】

一方で、ステップS21において、時刻が重複していないと判定された場合、又は、時刻が一致していたとしてもステップS22においてサブユニットが一致していないと判定された場合、予約のダブルブッキングが発生する恐れはないので、コントローラ11は、ステップS23の処理として、変数*i*を1インクリメントした後、ステップS19に戻り、*number_of_entries*から変数*i*を減算した値が0より大きくないと判定されるまで、同様の処理を繰り返し実行する。すなわち、コントローラ11では、DVCR3BBS34のRSB61に記憶されている全ての*object entry [i]*について、重複する時刻の予約がなされているか否かの検索を行う。

【0101】

また、ステップS19において、*number_of_entries*から変数*i*を減算した値が”0”より大きくないと判定された場合（全ての*object entry [i]*の検索が終了した場合）、コントローラ11は、図20のステップS25の処理として、前記図23に示したクリエイトコマンドをバス2を介してDVCR3のBBS34（実際にはBBS34を制御するコントローラ31）へ送り、当該BBS34のRSB61の*object entry*をクリエイトする。

【0102】

次に、コントローラ11は、ステップS26の処理として、前記図27に示したライトディスクリプタコマンドをバス2を介してDVCR3のBBS34（実際にはBBS34を制御するコントローラ31）へ送り、当該DVCR3のBBS34のRSB61の*entry_specific_information fields*の部分（図2、図7）に予約内容を書き込む。これにより、*start_time*, *Duration*, *repeat_information*, 使用するサブユニット (*subunit_type_and_ID*) などが書き込まれる。

【0103】

次に、コントローラ11は、ステップS27の判定処理において、DVCR3のBBS34内のRSB61の図7に示したResource Schedule Entryに、テキスト情報として予約の設定内容を追加すると判定した場合、ステップS28の処理として、図28に示したSubunit Identifier Descriptor内のRSBボードタイプを読み出し、次のステップS29にて、BBS34のRSB61のバージョンが1.0、すなわちRSBのResource Schedule Entryにテキスト情報を書き込むためのinformation blockとして、図7及び図14～図17に示したcharacter_code_information_blockと、language_code_information_blockと、raw_text_information_blockとが定義されているバージョン1.0よりも、RSBのバーションが大きいか否か判定する。

【0104】

ステップS29にてバージョン1.0より大きいと判定してステップS30に進むと、コントローラ11は、ユーザからの予約設定情報の入力に応じたテキスト情報を、上記RSB61のResource Schedule Entryのinformation blockに追加する。

【0105】

次に、ステップS31の処理に進むと、コントローラ11は、図31に示したクローズコマンドをバス2を介してDVCR3のBBS34（実際にはBBS34を制御するコントローラ31）へ送り、当該BBS34のRSB61をクローズする。

【0106】

その後ステップS32に進み、コントローラ11は、予約に関連する他のリソースが存在するか否かを判定する。この場合、これまでの処理によりIRD1のチューナサブユニット12とDVCR3のVCRサブユニット33についての予約の処理は終了しており、他に予約処理を行うべきサブユニットがないため、処理を終了する。

【0107】

次に、オブジェクトID設定処理について、図32のフローチャートを参照し

て説明する。この処理は、ユーザがIRD1に対して、DVCR3を用いる録画予約の操作を入力し、その操作がコントローラ11に検知され、上述した図18乃至図20の予約処理が実行されて、その録画予約が認められた後、開始される。この例においては、所定の時刻から所定の時間の間、実行されるイベント（予約処理）に関し、それを識別するために、オブジェクトIDが対応される。このオブジェクトIDは、72ビットで構成され、そのうちのMSB側の64ビットは、その予約に関する全ての情報を持っている機器の固有のID、具体的にはその機器のGUID (Global Unique ID)とされ、LSB側の8ビットは、その機器内で設定される固有の値とされ、具体的にはレコードID (record ID) とされる。このようにオブジェクトIDをGUIDとレコードIDとで構成することにより、イベントを識別するためのオブジェクトIDを設定すると、ID設定の処理が容易となる。

【0108】

すなわちこのオブジェクトIDは、バス2に接続されている各機器の間において、識別可能である必要がある。そうでなければ、この例においては、1つのユニット（機器）のRSBの内容を他のユニット（機器）が読み込み可能であり、複数のユニットのサブユニットにおいて、共同して（関連して）処理が実行されることがあるので、関連する処理があるか否かを、他のユニットが識別できる必要があるからである。

【0109】

バス2に接続されている、ユニット間において、識別可能なものであれば、例えば、オブジェクトIDをイベントが発生する毎に、番号1から順番に順次設定するようにすることも可能である。しかしながらそのようにすると、所定のユニットが、所定のイベントに関して、オブジェクトIDを設定しようとした場合、バス2に接続されている全てのユニットが、既に設定しているオブジェクトIDを読み出して、競合するものがあるか否かを判定し、競合しないものに設定する必要が生じる。しかしながらこの例においては、GUIDがオブジェクトIDに含まれており、GUIDの値は、ユニット毎に異なるものであることが保証されている。したがって各ユニットは、レコードIDを自分自身の内部において、競

合するものがないように設定すれば、GUIDとレコードIDを組み合わせて得られるオブジェクトIDは、他のユニットが設定したオブジェクトIDと競合することはない。そこで各ユニットは、図32のフローチャートに示すような処理を行って、RSBに登録するイベントに対するオブジェクトIDを設定する。

【0110】

ステップS41において、コントローラ11は、オブジェクトID（=Global Unique ID+record ID）のうちのレコードIDとして、IRD1内において、その予約情報を一意に識別させるための仮IDを発生する。

【0111】

ステップS42において、コントローラ11は、IRD1のBBS14内のRSBに登録されているイベントの中から、1つのイベントを抽出して、そのイベントに対応するオブジェクトIDの中のレコードIDを抽出する。ステップS43において、コントローラ11は、ステップS41で発生した仮IDと、ステップS42で抽出したレコードIDが一致するか否かを判定し、一致する場合、ステップS41に戻り、仮IDを変更し、一致しない仮IDが発生されるまで、ステップS41乃至S43の処理を繰り返す。

【0112】

仮IDとステップS42で抽出したレコードIDが一致しないと判定された場合、ステップS44において、コントローラ11は、関連するサブユニットに関するRSBに公開されている全てのイベントを抽出したか否かを判定し、まだ抽出していないイベントが残っている場合はステップS42に戻り、全てのイベントを読み出したと判定するまで、ステップS42乃至S44の処理を繰り返す。全てのイベントを読み出したと判定された場合、ステップS45において、コントローラ11は、IRD1の機器ID（GUID）に、ステップS41で発生した仮ID（レコードID）を付加して、オブジェクトIDを生成し、RSBに記述する。

【0113】

このように、オブジェクトIDは、関連するサブユニットに関するRSBをチェックするだけで（関連しないサブユニットを有するユニットのRSBをチェック

クすることなく）設定することができる。これは、上述したように、他のユニットのRSBにおけるオブジェクトIDは、そのユニットのGUIDが含まれているため、他のユニットが設定したオブジェクトIDは、自分自身が設定したオブジェクトIDと一致することが有り得ないからである。したがって、オブジェクトIDを簡単に設定することが可能となる。

【0114】

本実施の形態において、上述した一連の処理は、ハードウェアにより実行させることもできるが、ソフトウェアにより実行させることもできる。一連の処理をソフトウェアにより実行させる場合には、そのソフトウェアを構成するプログラムが、専用のハードウェアとしてのコントローラに組み込まれているか若しくは当該プログラムがインストールされた、例えば汎用のパーソナルコンピュータなどにより各種の機能を実行することが可能となる。

【0115】

汎用のパーソナルコンピュータ101は、例えば、図33に示すように、CPU (Central Processing Unit) 111を内蔵している。CPU111には、バス115を介して入出力インターフェース116が接続されており、CPU111は、入出力インターフェース116を介して、ユーザから、キーボード、マウスなどよりなる入力部118から指令が入力されると、それに対応して、ROM (Read Only Memory) 112あるいはハードディスク114などの記録媒体、または、ドライブ120に装着された磁気ディスク131、光ディスク132、光磁気ディスク133などの記録媒体から、それらに記録されている、上述した一連の処理を実行するプログラムを読み出し、RAM (Random Access Memory) 113に書き込み、実行する。なお、ハードディスク114に格納されているプログラムには、予め格納されてユーザに配布されるものだけでなく、衛星もしくは、ネットワークから転送され、通信部119により受信されてダウンロードされたプログラムも含まれる。

【0116】

また、CPU111は、プログラムの処理結果のうち、画像信号を、入出力インターフェース116を介して、LCD (Liquid Crystal Display), CRT

(Cathode Ray Tube) などよりなる表示部117に出力する。

【0117】

なお、本発明は、上述の説明した実施の形態に限定されるものではなく、本発明の精神を逸脱することなく種々の変形が可能とされるものである。例えば本実施の形態では、IRD1のコントローラ11が、DVCR3のBBS34のRSB61への前記テキスト情報の書き込みと読み出しを制御しているが、さらに他の機器（ユニット）のコントローラが当該DVCR3のBBS34のRSB61へのテキスト情報の書き込みと読み出しを制御することも可能である。このとき、本実施の形態によれば、例えばDCVR等で確認画面を出すような場合に、予約の内容をRSBのinformation blockの形式としているので、他の機器による予約の概要を、上記DVCRの確認画面上で、本体装置や外部装置の区別無く、同じ情報量でユーザに表示することができる。

【0118】

【発明の効果】

本発明の情報処理装置及び方法、媒体においては、機能実行手段の使用予定に関する第1の情報に含まれない第2の情報を入手し、第2の情報を所定のブロック形式とし、第1の情報の付加情報として格納手段に格納させる制御を行うことにより、ユーザや各機器が予約情報等の詳細を入力でき、また、各機器が他の機器による予約情報等の詳細入手でき、及び、各機器が他の機器に対して自己が持つ予約情報等の詳細を知らせることを実現可能となり、その結果、ダブルブッキングの発生を抑止可能となるだけでなく、分かり易く且つ使い易いネットワークシステムを構築可能である。

【0119】

すなわち、本発明によれば、例えば予約のダブルブッキングが起きるときの警告表示を出すときに、そのダブルブッキングの発生原因をより分かり易い形でユーザに知らせることができる。また、本発明によれば、例えば仮の予約を可能とし、仮予約なので上書きしても良いという情報をエントリ時に書き込むことにより、後から予約するユーザがより適切な行動、つまり仮予約されていたものを消して新たな予約を行ったりする等の行動を取ることができるようにになる。さらに

、例えばDCVR等で確認画面を出す場合は、その予約の概要を表示することが一般に行われるが、本発明によれば、予約の内容（第2の情報）を所定のブロック形式とすることで、他の機器による予約の概要を、確認画面上で本体装置、外部装置の区別無く、同じ情報量でユーザに表示することができる。

【図面の簡単な説明】

【図1】

本発明実施の形態のネットワークシステムの構成例を示すブロック図である。

【図2】

ターゲットデバイスのBBSのRSBをポスティングデバイスが制御する際の説明に用いる図である。

【図3】

BBSのRSBのフォーマットを説明する図である。

【図4】

図3のWrite Enabled list_specific_informationのフォーマットを説明する図である。

【図5】

図4のboard_typeのフォーマットを説明する図である。

【図6】

図3のobject_entryのフォーマットを説明する図である。

【図7】

図6のResource Schedule Entryのフォーマットを説明する図である。

【図8】

図7のstart_timeのフォーマットを説明する図である。

【図9】

図7のDurationのフォーマットを説明する図である。

【図10】

図7のrepeat_typeのフォーマットを説明する図である。

【図11】

スケジュールが週毎に繰り返される場合のrepeat_informationのフォーマット

を説明する図である。

【図12】

スケジュールが所定の間隔で行われる場合のrepeat_informationのフォーマットを説明する図である。

【図13】

図7のInfo blocksのフォーマットを説明する図である。

【図14】

図7に示すResource Schedule Entryのcharacter_code_information_blockのフォーマットを説明する図である。

【図15】

図7に示すResource Schedule Entryのlanguage_code_information_blockのフォーマットを説明する図である。

【図16】

図7に示すResource Schedule Entryのraw_text_information_blockのフォーマットを説明する図である。

【図17】

図16に示すraw_text_information_block内に記述されるraw_text_dataの一例を説明する図である。

【図18】

図1のネットワークシステムの動作のうち、使用したい機器のRSBが書き込み可能かどうか調べる手順を説明するフローチャートである。

【図19】

図1のネットワークシステムの動作のうち、使用予定のリソースが他の機器から使われる予定になっているかどうか確認する手順を説明するフローチャートである。

【図20】

図1のネットワークシステムの動作のうち、使用予定を書き込む時の手順を説明するフローチャートである。

【図21】

ライトオープンコマンドのフォーマットを説明する図である。

【図22】

リードコマンドのフォーマットを説明する図である。

【図23】

クリエイトコマンドのフォーマットを説明する図である。

【図24】

図12のsubfunction_1を説明する図である。

【図25】

図24のsubfunction_1の詳細を説明する図である。

【図26】

図25におけるフィールドの値の例を示す図である。

【図27】

ライトディスクリプタコマンドのフォーマットを説明する図である。

【図28】

BBSのフォーマットを説明する図である。

【図29】

BBSのroot_object_list_IDを説明する図である。

【図30】

図28のBBSのsupported_board_type_specific_informationのフォーマットを説明する図である。

【図31】

クローズコマンドのフォーマットを説明する図である。

【図32】

オブジェクトID設定処理を説明するフローチャートである。

【図33】

コンピュータの構成例を示すブロック図である。

【図34】

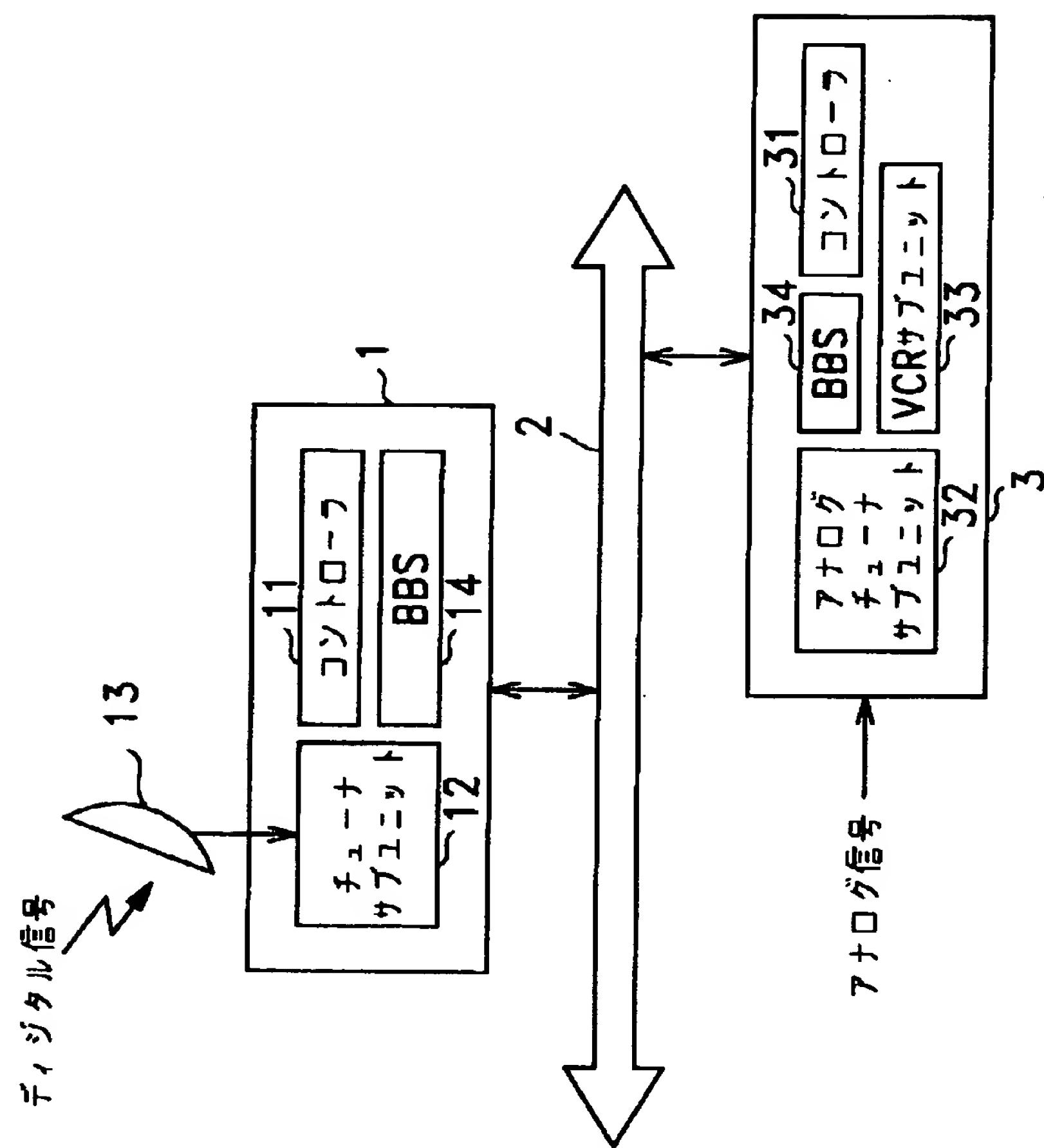
従来のネットワークシステムの構成例を示すブロック図である。

【符号の説明】

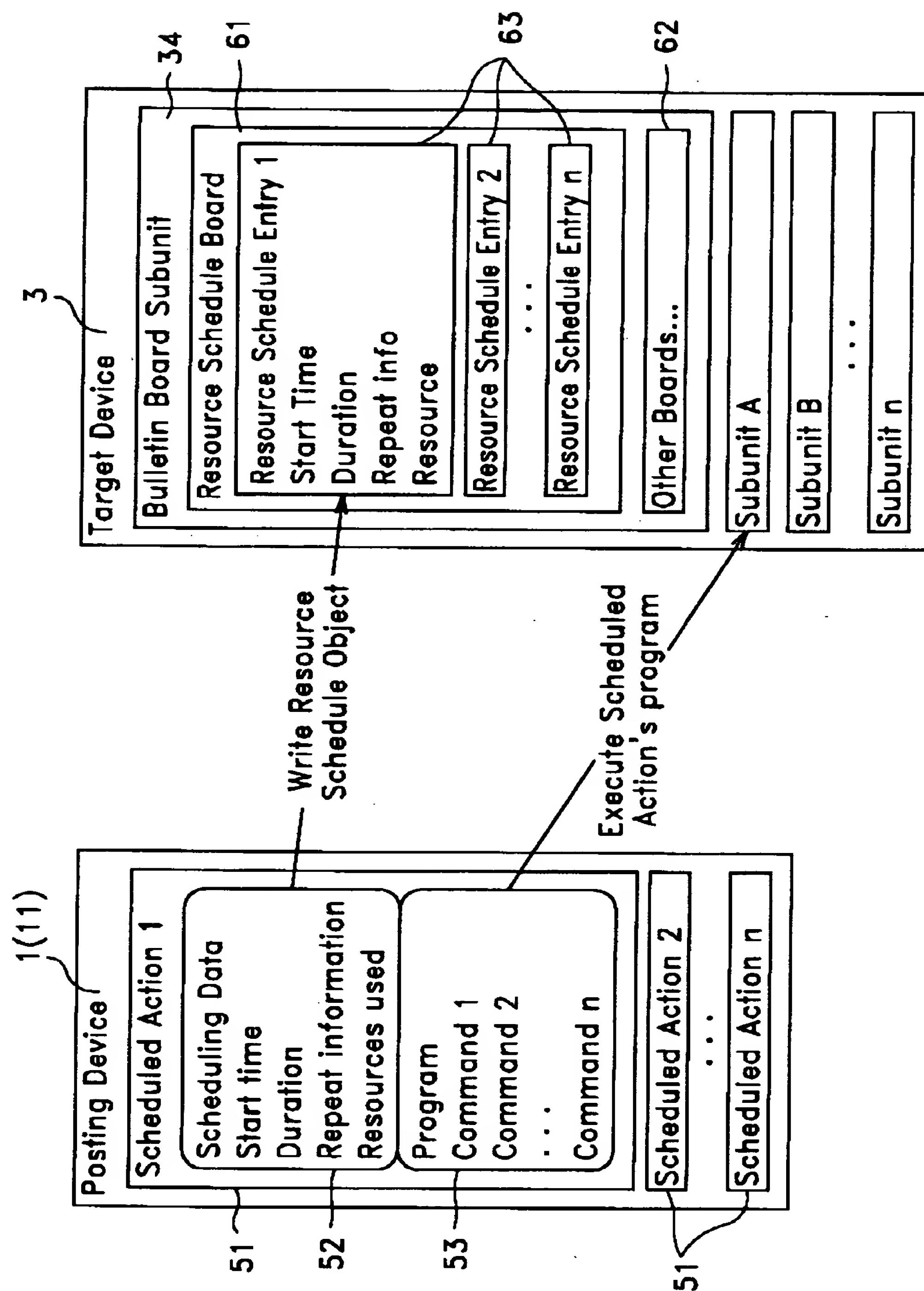
1 I R D、 2 I E E E 1 3 9 4 シリアルデータバス、 3 D V C R、
1 1 コントローラ、 1 2 チューナサブユニット、 1 4 B B S、 3
1 コントローラ、 3 2 アナログチューナサブユニット、 3 3 V C Rサ
ブユニット、 3 4 B B S、 6 1 R S B

【書類名】 図面

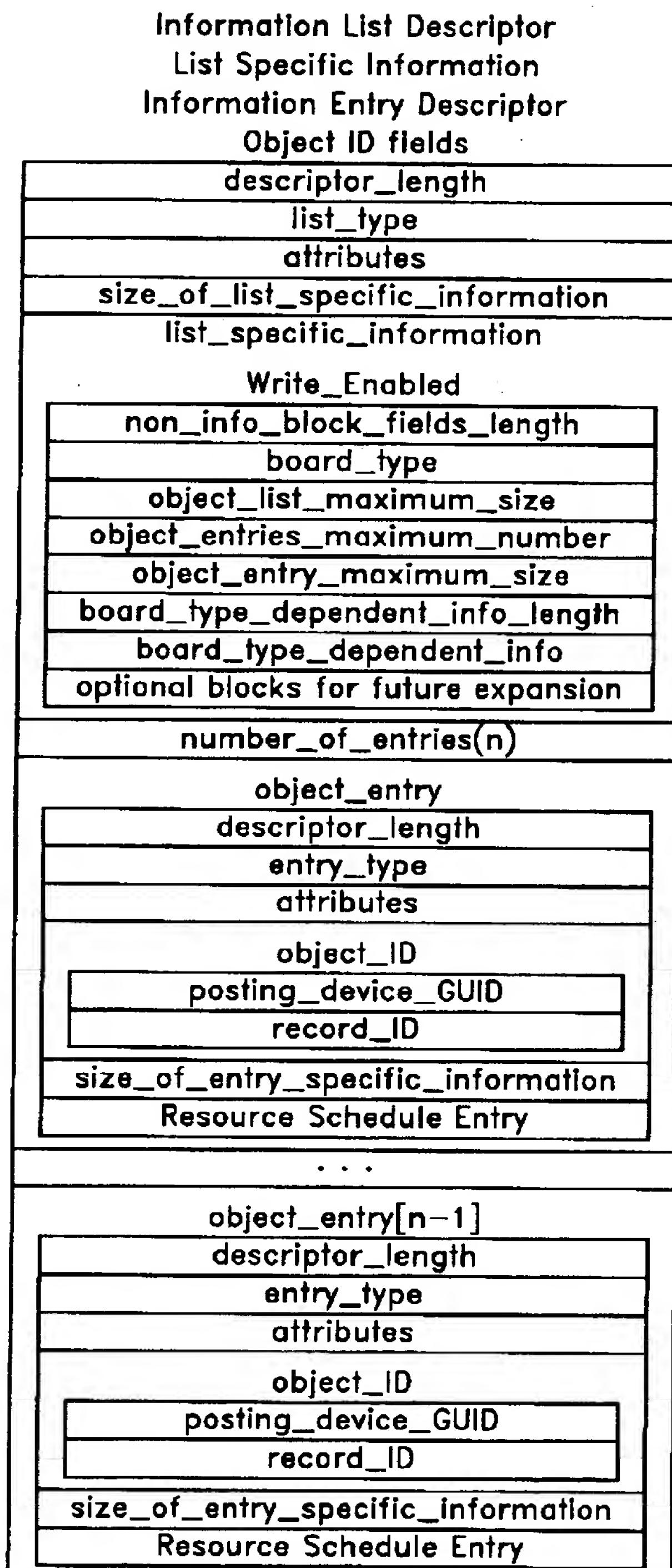
【図1】



【図2】



【図3】



【図4】

Address_offset	Contents
00 ₁₆	non_info_block_fields_length
01 ₁₆	
02 ₁₆	board_type
03 ₁₆	object_list_maximum_size
04 ₁₆	
05 ₁₆	object_entries_maximum_number
06 ₁₆	
07 ₁₆	object_entry_maximum_size
08 ₁₆	
09 ₁₆	board_type_dependent_information_length
0A ₁₆	
0B ₁₆	board_type_dependent_information
0C ₁₆	
0D ₁₆	
:	optional info blocks for future expansion
:	
:	
:	

Write enabled list_specific_information fields for the Information List
Descriptor

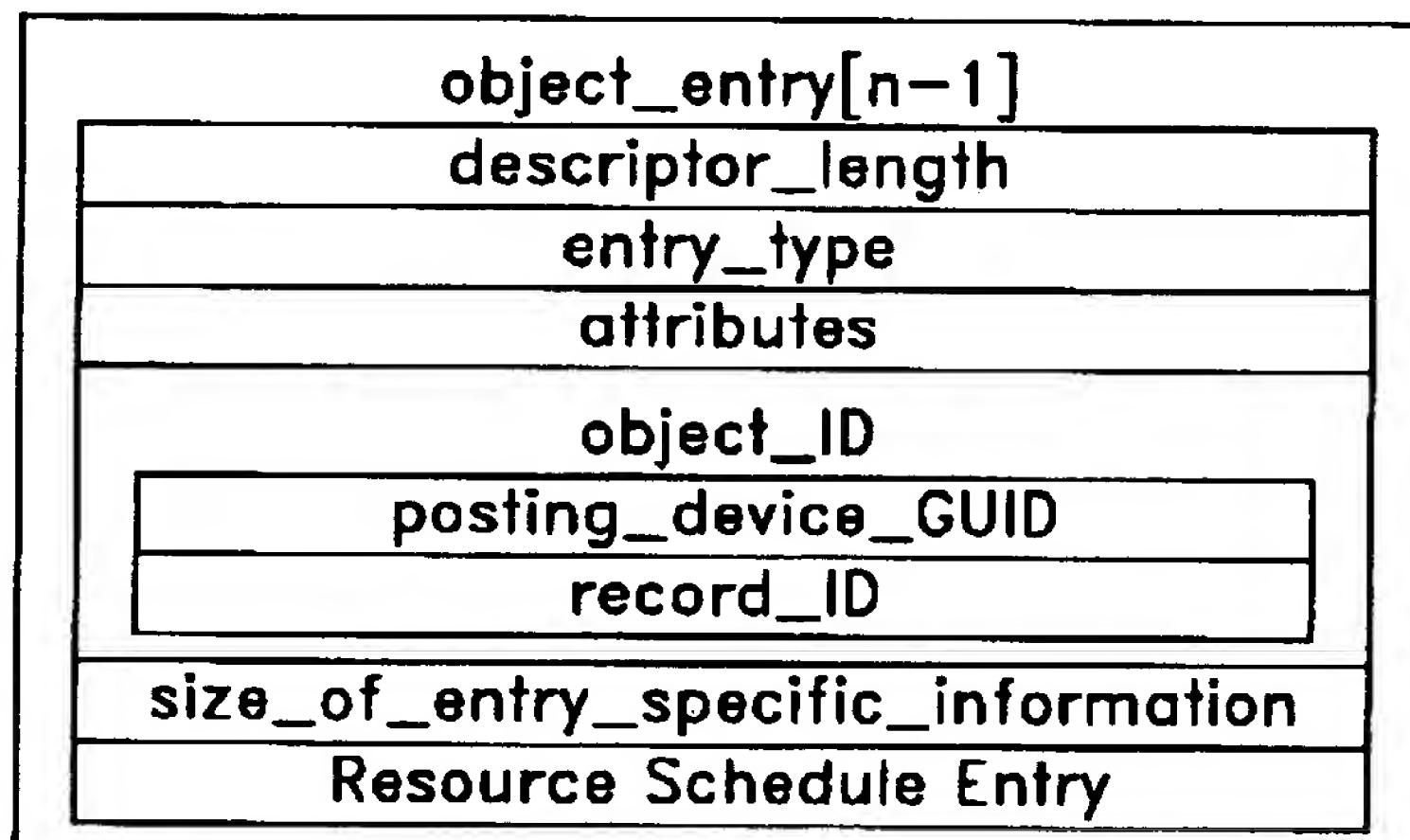
【図5】

Existing board types

Value	Board type
00 ₁₆	Reserved
01 ₁₆	Resource Schedule Board
02 ₁₆ -FF ₁₆	Reserved for future specification

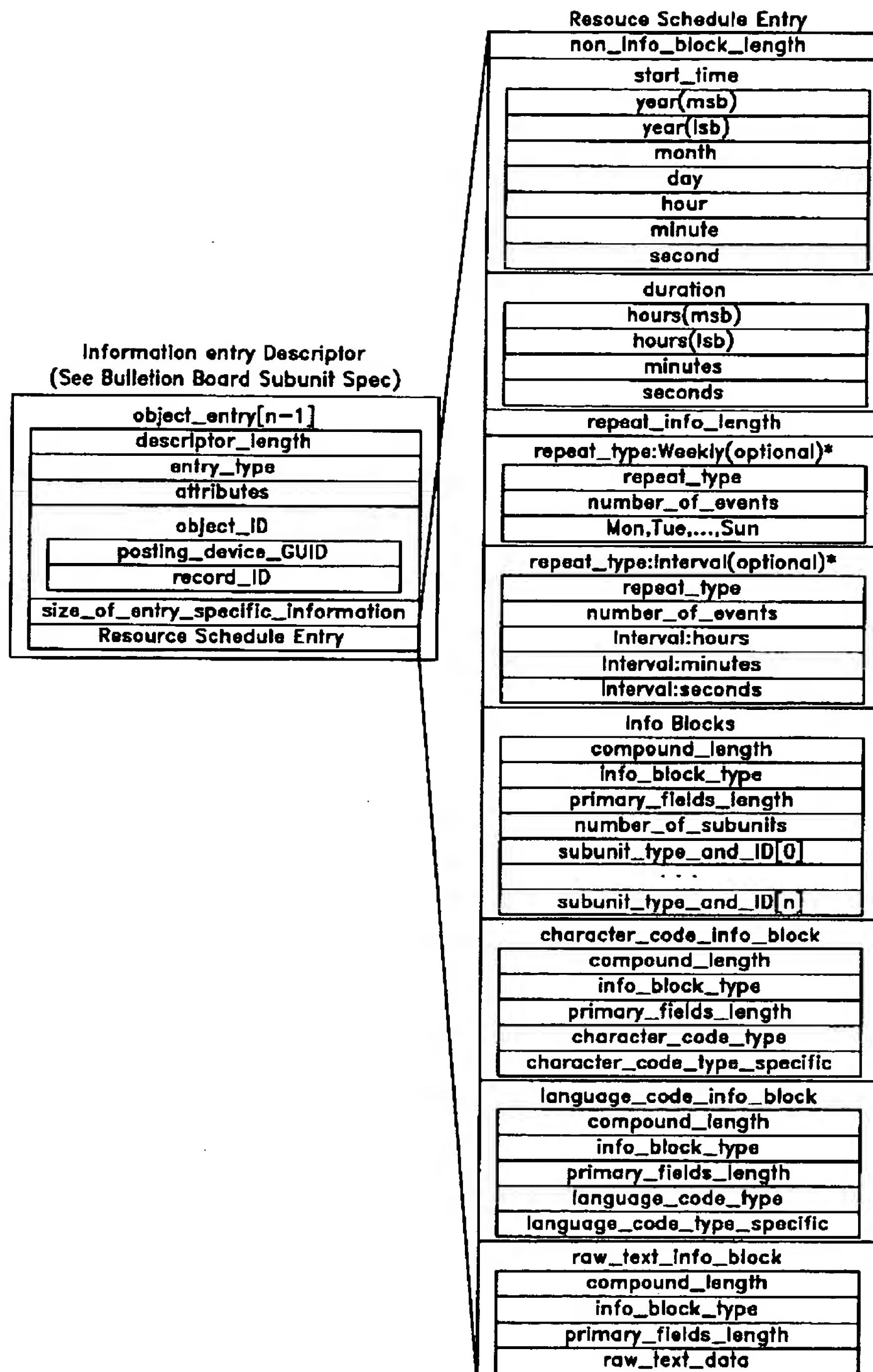
【図6】

Information Entry Descriptor



Resource Schedule Entry high level view

【図7】



【図8】

Address_offset	Contents
00 ₁₆	year(msb)
01 ₁₆	year(lsb)
02 ₁₆	month
03 ₁₆	day
04 ₁₆	hour
05 ₁₆	minute
06 ₁₆	second

start_time fields for Resource Schedule Entries

【図9】

Address_offset	Contents	
00 ₁₆	Reserved(4 bits)	hours(msb)
01 ₁₆		hours(lsb)
02 ₁₆		minutes
03 ₁₆		seconds

duration fields for Resource Schedule Entries

【図10】

repeat_type value assignment

Values	definition
00 ₁₆	Weekly schedule
01 ₁₆ -0F ₁₆	reserved
10 ₁₆	Interval schedule
0F ₁₆ -FF ₁₆	reserved

【図11】

address_offset	contents	msb						lsb
OE_{16}	repeat_type							
OF_{16}	number_of_events							
10_{16}	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Reserved

repeat_information fields for Weekly

【図12】

address_offset	contents
OE_{16}	repeat_type
OF_{16}	number_of_events
10_{16}	Reserved(4 bits) Interval:hours(msb)
11_{16}	interval:hours(lsb)
12_{16}	interval:minutes
13_{16}	interval:seconds

repeat_information fields for interval

【図13】

address_offset	contents
00 ₁₆	compound_length
01 ₁₆	
02 ₁₆	info_block_type
03 ₁₆	
04 ₁₆	primary_fields_length
05 ₁₆	
06 ₁₆	number_of_subunits
07 ₁₆	subunit_type_and_ID[0]
:	:

Subunit_Resource_info_block

【図14】

character_code_info_block	
Address_offset	Contents
00 00 ₁₆	compound_length
00 01 ₁₆	info_block_type=00 08 ₁₆ (character_code_info_block)
00 02 ₁₆	primary_fields_length
00 03 ₁₆	character_code_type
00 04 ₁₆	character_code_type_specific
00 05 ₁₆	
00 06 ₁₆	
00 07 ₁₆	
	..

【図15】

<code>language_code_info_block</code>	
<code>Address_offset</code>	<code>Contents</code>
00 00 ₁₆	
00 01 ₁₆	
00 02 ₁₆	<code>info_block_type=00 09₁₆(language_code_info_block)</code>
00 03 ₁₆	
00 04 ₁₆	<code>primary_fields_length</code>
00 05 ₁₆	
00 06 ₁₆	<code>language_code_type</code>
00 07 ₁₆	<code>language_code_type_specific</code>
:	
..	

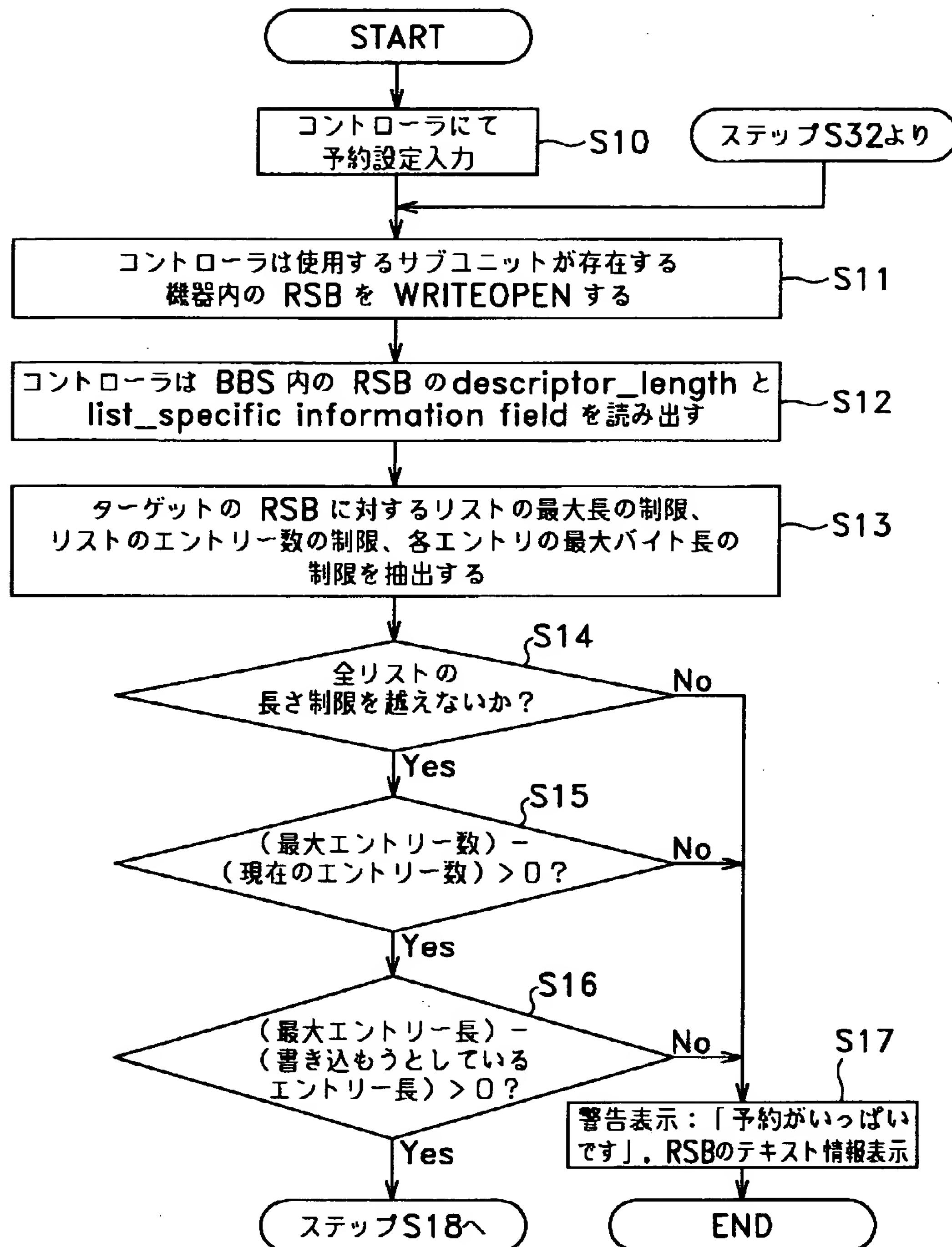
【図16】

Address_offset	raw_text_info_block	Contents
00 00 ₁₆		compound_length
00 01 ₁₆		info_block_type=00 0A ₁₆ (raw_text_info_block)
00 02 ₁₆		primary_fields_length
00 03 ₁₆		
00 04 ₁₆		
00 05 ₁₆		
00 06 ₁₆		
:		
		raw_text_data
:		

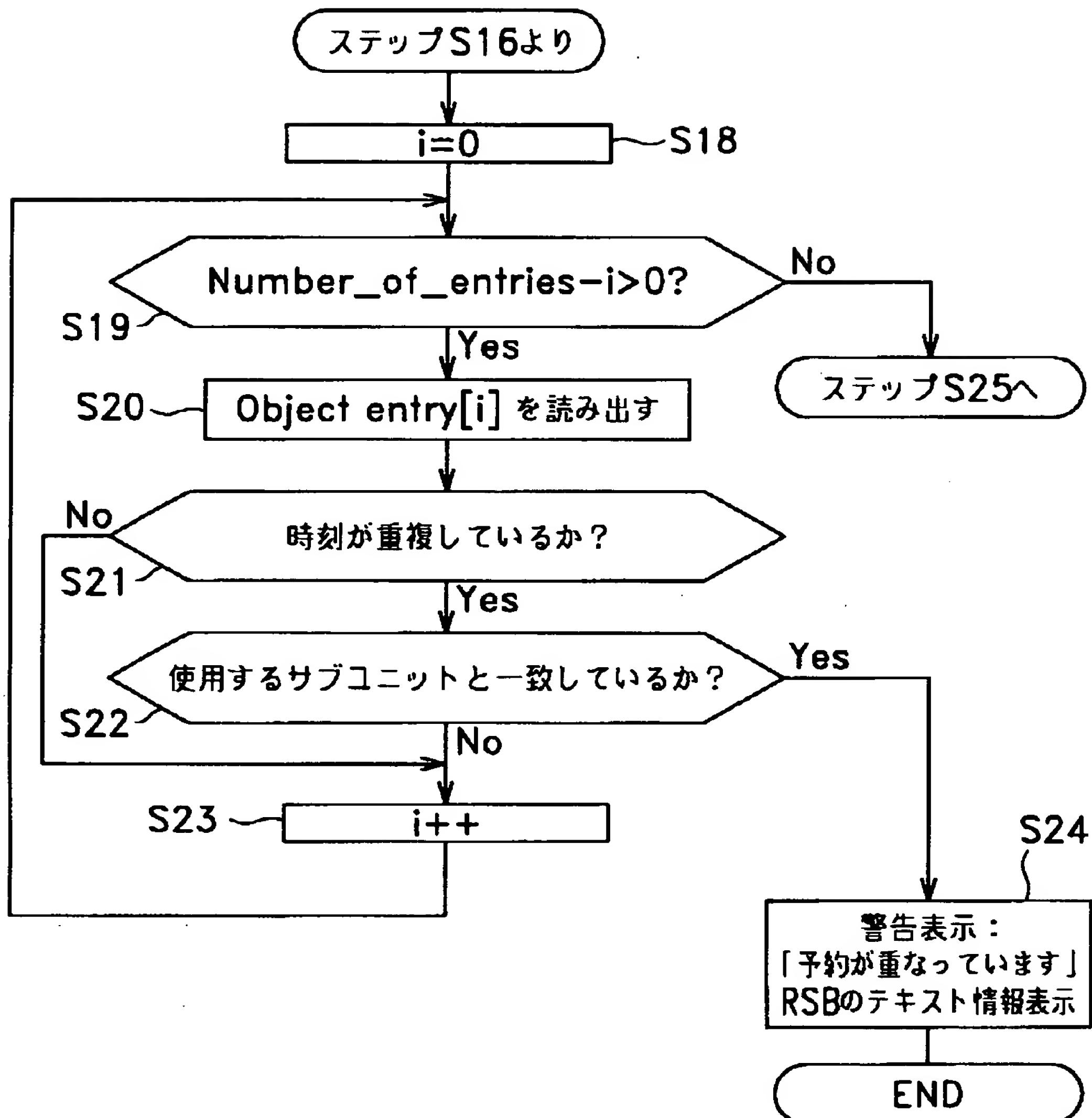
【図17】

raw_text_data
チャンネル
プログラム名（番組）
制御情報（再生、録画、停止等）
備考（ペイバー・ビュー）
プロバイダ名
仮予約中

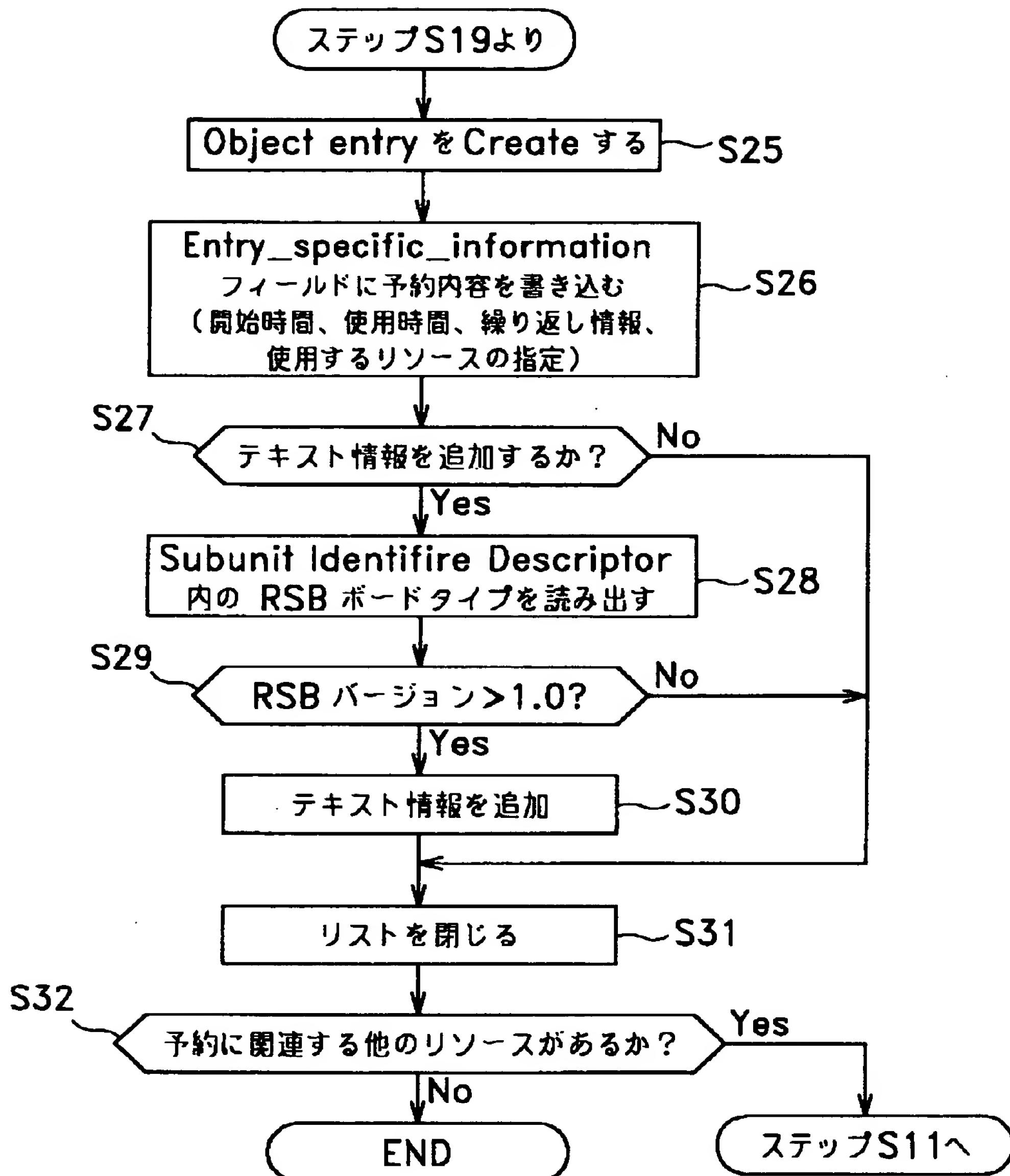
【図18】



【図19】



【図20】



【図21】

WRITE OPEN コマンド

opcode	OPEN DESCRIPTOR
operand 0	descriptor_type=10 ₁₆
operand 1	List ID:00 ₁₆
operand 2	List ID:01 ₁₆
operand 3	subfunction WRITE OPEN 03 ₁₆
operand 4	reserved 00 ₁₆

【図22】

READ コマンド

opcode	msb						lsb
opcode							READ DESCRIPTOR (09 ₁₆)
operand 0							descriptor identifier
operand 1							:
:							:
:							read_result_status
:							reserved : 00 ₁₆
:							data_length
:							address

【図23】

	msb						lsb
opcode	CREATE DESCRIPTOR (00₁₆)						
operand 0	result						
operand 1	subfunction_1						
operand 2	reserved						
operand 3	subfunction_1_specification						
:							
:							

【図24】

subfunction_1の値	meaning
00₁₆	create a new descriptor
01₁₆	create a new object and its child list
all other values	reserved for future specification

【図25】

subfunction_1_specification for subfunction_1=01 ₁₆							
	msb						lsb
operand 3							
:	descriptor_identifier_where						
:							
:	descriptor_identifier_what_1						
:							
:	descriptor_identifier_what_2						
:							

【図26】

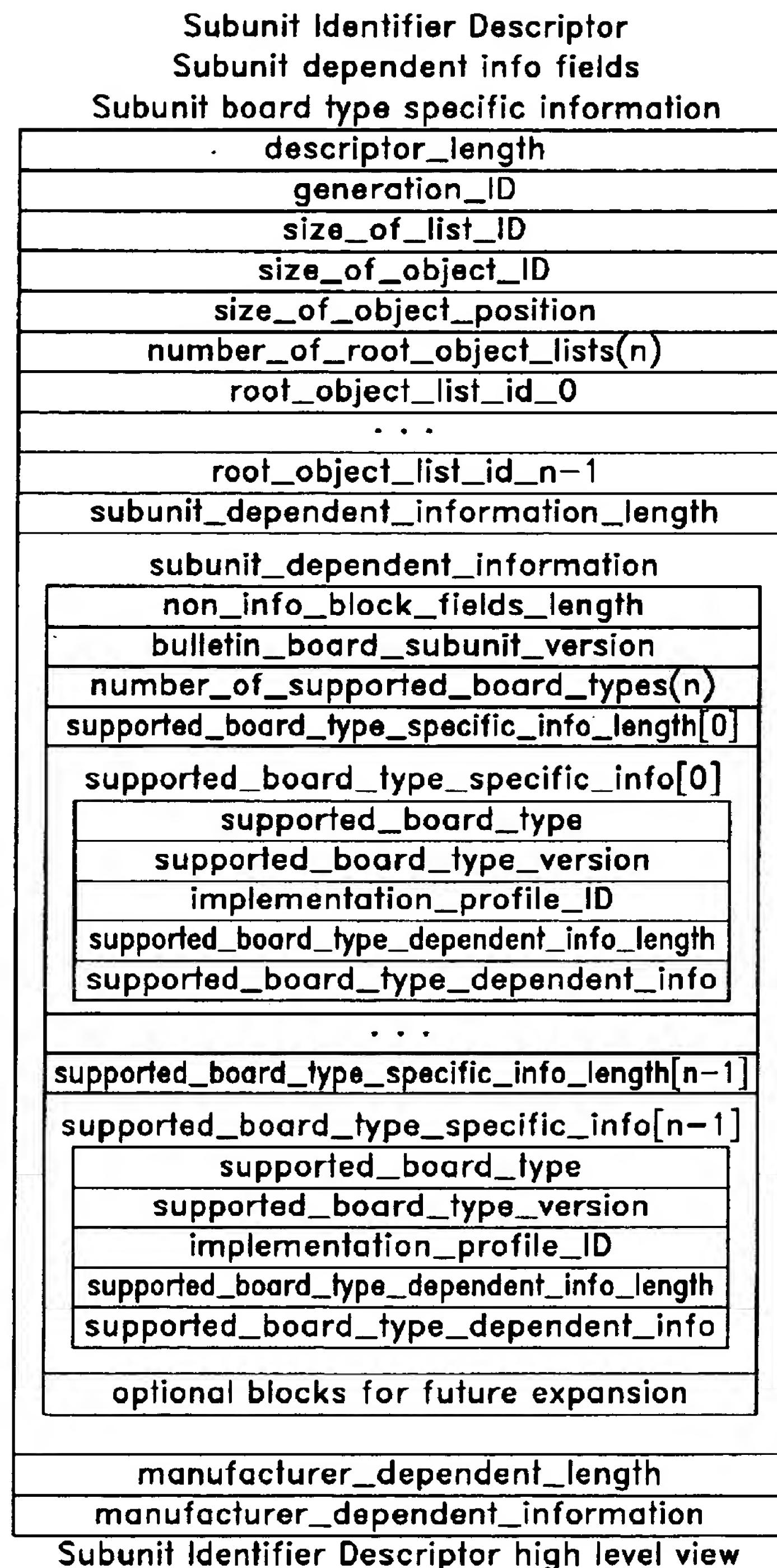
descriptor_type of descriptor_identifier_where	descriptor_type of descriptor_identifier_what_1	descriptor_type of descriptor_identifier_what_2	meaning
20₁₆	22₁₆	11₁₆	Create an object and its child list. create the new object and place it in the location specified by where, the entry_type is specified by what_1. Also create the new list as the child of the new object. The list_type is specified by what_2.
		all other values	reserved for future specification

【図27】

WRITE DESCRIPTOR コマンド

opcode	WRITE DESCRIPTOR ($0A_{16}$)
operand 0	descriptor identifier
:	subfunction:partial_replace(50_{16})
:	group_tag:immediate(00_{16})
:	replacement_data_length
:	address
:	original_data_length
:	replacement_data

【図28】



【図29】

root_object_list_ID Value Assignment

Value	List definition
1001_{16}	Resource Schedule List
$1002\text{--}10FF_{16}$	reserved

【図30】

Address_offset	Contents
00 ₁₆	supported_board_type
01 ₁₆	supported_board_type_version
02 ₁₆	implementation_profile_ID
03 ₁₆	supported_board_type_dependent_information_length
04 ₁₆	
05 ₁₆	supported_board_type_dependent_information
:	:
:	:

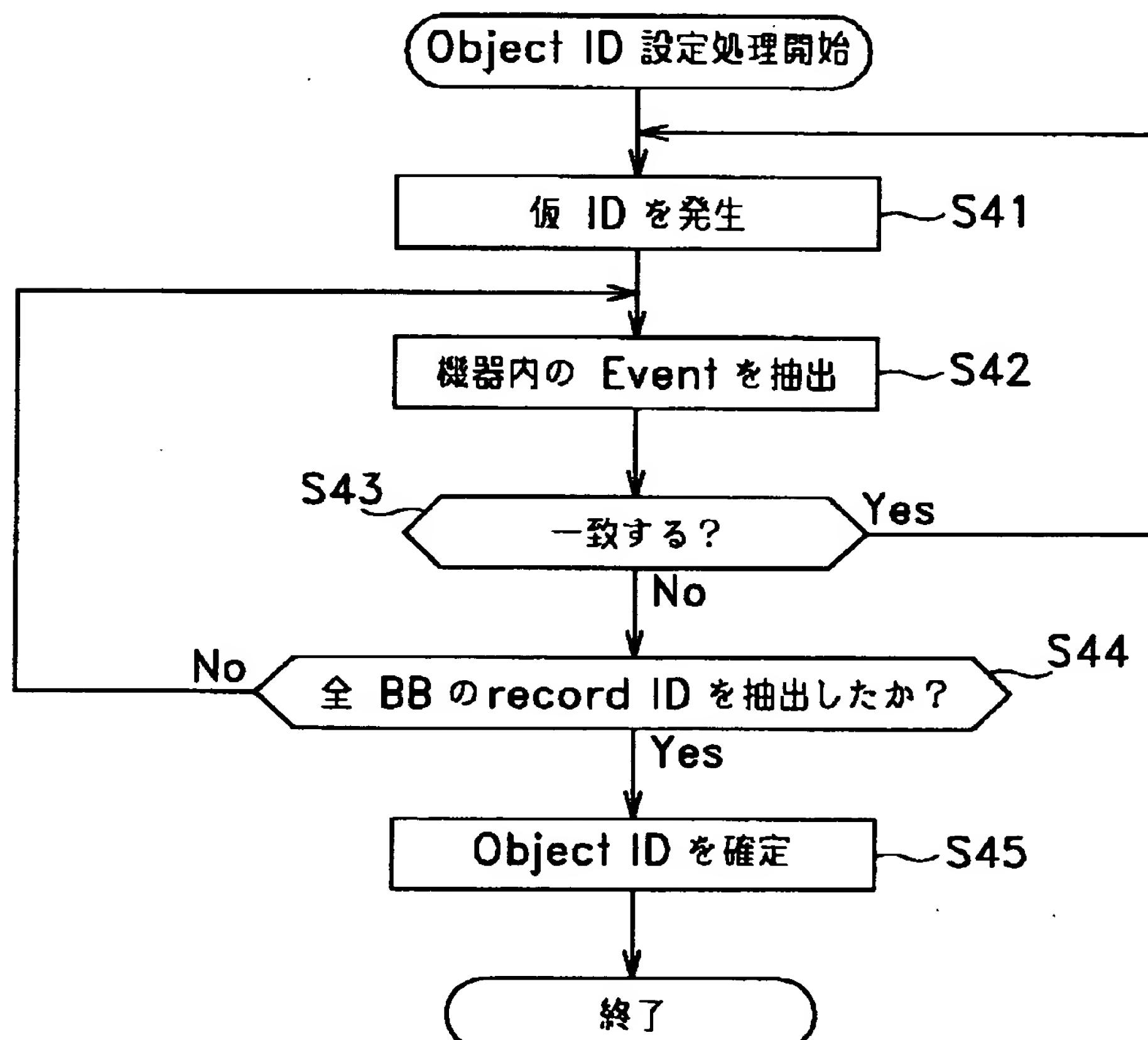
supported_board_type_specific_information fields

【図31】

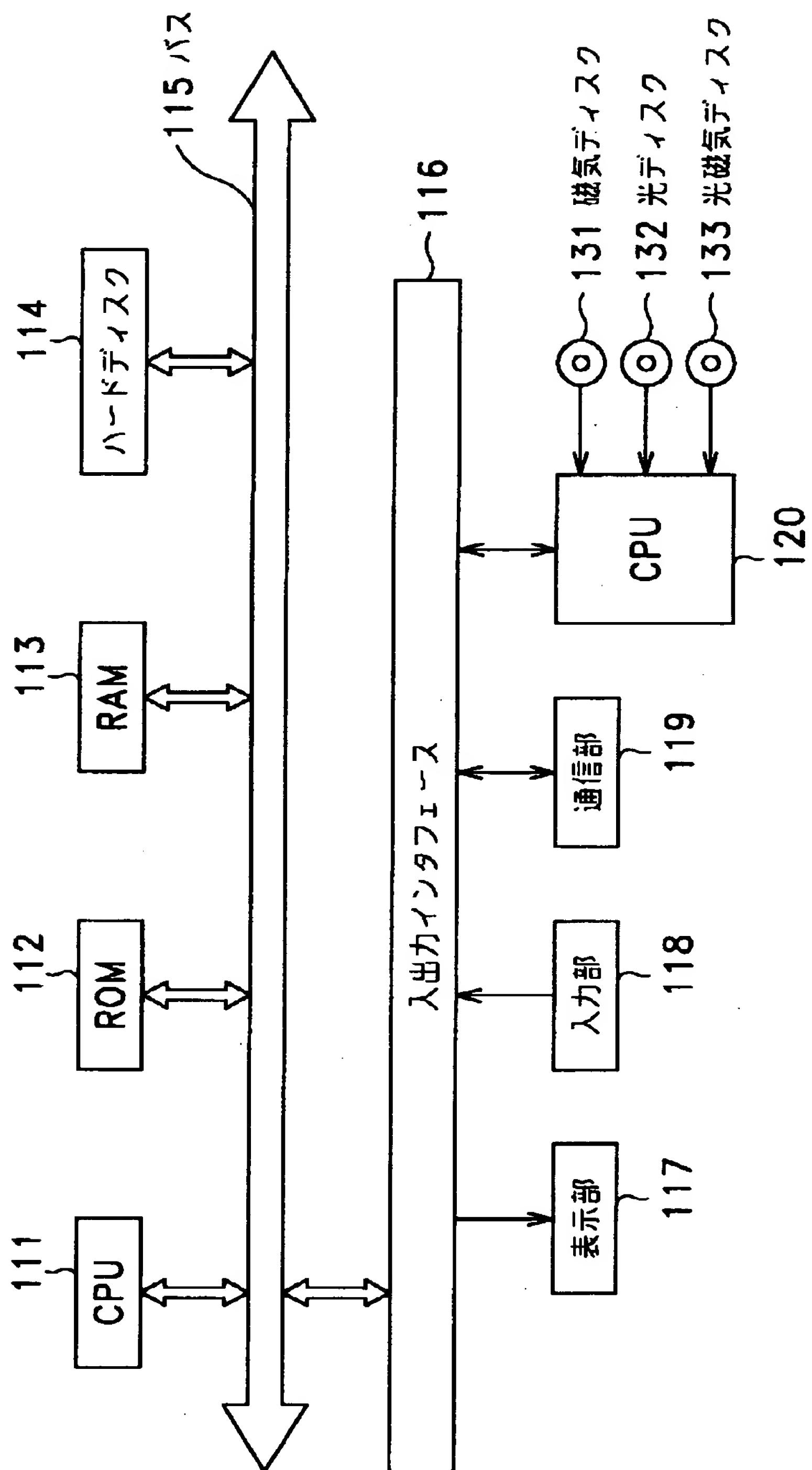
CLOSE コマンド

opcode	OPEN DESCRIPTOR
operand 0	descriptor type= 10_{16}
operand 1	List ID: 00_{16}
operand 2	List ID: 01_{16}
operand 3	subfunction CLOSE 00_{16}
operand 4	reserved 00_{16}

【図32】

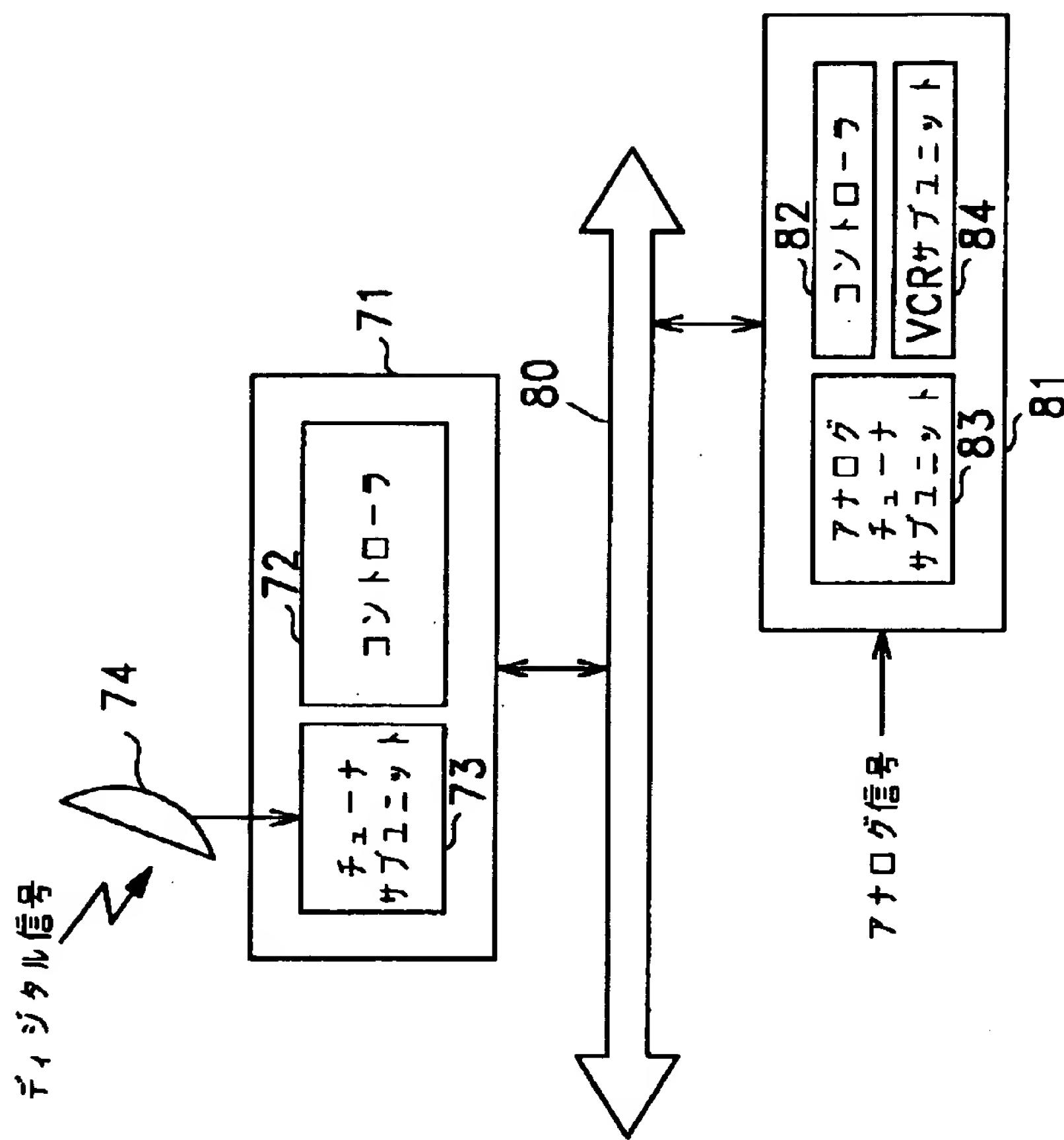


【図33】



パーソナルコンピュータ 101

【図34】



【書類名】 要約書

【要約】

【課題】 ユーザや各機器が予約情報等の詳細を入力でき、各機器が他の機器による予約情報等の詳細を入手でき、各機器が他の機器に対して自己が持つ予約情報等の詳細を知らせる手段を実現し、ダブルブッキングの発生を抑止し、分かり易く且つ使い易いネットワークシステムを構築可能とする。

【解決手段】 RSBのResource Schedule Entryに、予約の詳細を表すローテキストを含むraw_text_information_blockと、ローテキストのコーディングを表すcharacter_code_information_blockとlanguage_code_information_blockを記述する。

【選択図】 図7

出願人履歴情報

識別番号 [000002185]

1. 変更年月日 1990年 8月30日

[変更理由] 新規登録

住 所 東京都品川区北品川6丁目7番35号

氏 名 ソニー株式会社